COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

12/2013 VOL.56 NO.12

Data Science and Prediction

Making the Web Faster with HTTP 2.0

From MOOCs to SPOCs

The Lensless Camera

Software at Scale

ACM Books to Launch

Association for Computing Machinery

Take a look at the

Internet's future.

Find **the leaders** in data communication and networking from industry and academia, **all in one place, once a year.**

The **Annual Festival** of the **ACM** Special Interest Group on **Data Communication**

Chicago August 17 - 22, 2014

Association for Computing Machinery

acm sigcomm

conferences.sigcomm.org/sigcomm/2014

tvx 2014

ACM INTERNATIONAL CONFERENCE ON INTERACTIVE EXPERIENCES FOR TELEVISION AND ONLINE VIDEO

25-27 JUNE, 2014
NEWCASTLE UPON TYNE
UK

Paper Submissions by
3 February 2014

Workshop, Demo, WIP
DC, Grand Challenge
& Industrial
Submissions by
31 March 2014

Welcoming Submissions on
Content Production
Systems & Infrastructures
Devices & Interaction Techniques
Experience Design & Evaluation
Media Studies
Data Science & Recommendations
Business Models & Marketing
Innovative Concepts & Media Art

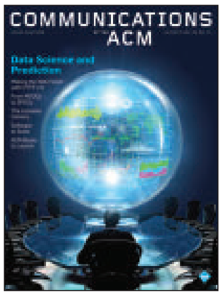
TVX2014.COM

COMMUNICATIONS OF THE ACM

Association for Computing Machinery
Advancing Computing as a Science & Profession

PHOTOGRAPH COURTESY OF ALCATEL-LUCENT, USA, INC.

12/2013
VOL. 56 NO. 12



About the Cover:
In this month's cover story (p. 64), Vasant Dhar examines ways to exploit the predictive power of big data for "actionable insight" that computers and human users can tap for fact-based decision making.
Cover photo illustration by Barry Downard.

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

STAFF

DIRECTOR OF GROUP PUBLISHING
Scott E. Delman
publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Larry Fisher
Web Editor
David Roman
Editorial Assistant
Zarina Strakhan
Rights and Permissions
Deborah Cotton

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Directors
Mia Angelica Balaquiot
Brian Greenberg
Production Manager
Lynn D'Addesio
Director of Media Sales
Jennifer Ruzicka
Public Relations Coordinator
Virginia Gold
Publications Assistant
Emily Williams

Columnists
David Anderson; Phillip G. Armour; Michael Cusumano; Peter J. Denning; Mark Guzdial; Thomas Haigh; Peter Harsha; Leah Hoffmann; Mari Sako; Pamela Samuelson; Marshall Van Alstyne; Cameron Wilson

CONTACT POINTS

Copyright permission
permissions@cacm.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/guidelines

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

Director of Media Sales
Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmmediasales@acm.org

Association for Computing Machinery (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

Subscriptions
An annual subscription cost is included in ACM member dues of $99 ($40 of which is allocated to a subscription to *Communications*); for students, cost is included in $42 dues ($20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is $100.

ACM Media Advertising Policy
Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current Advertising Rates can be found by visiting http://www.acm-media.org or by contacting ACM Media Sales at (212) 626-0686.

Single Copies
Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Association for Computing Machinery

Printed in the U.S.A.

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/2541106

M. Tamer Özsu

ACM Books to Launch

The academic and technical publishing landscape has changed significantly over the past decade. Following a period of massive consolidation in the scientific, technical, and medical publishing industry, there are now only a handful of commercial houses publishing computer science research, and their attention to computer science books, and in particular to graduate textbooks and research monographs where sales are modest compared to the undergraduate market, has dwindled. There are certainly exceptions, and university presses have filled some of the gap, but in today's environment it is entirely accurate to say a considerable vacuum exists in scholarly computer science book publishing. As a result, ACM has decided to play a significant role in filling some of this void by launching a new book program called ACM Books that will enhance its already rich journal and conference publication portfolio. We will begin publishing books in 2014.

ACM Books will fill a unique space in the publishing domain by including books from across the entire spectrum of computer science subject matter. The series will initially focus on four kinds of books: graduate-level textbooks, deep research monographs that provide an overview of established and emerging fields, practitioner-level professional books, and books devoted to the history and social impact of computing. This publishing program will address the information needs of various members of the computing community, including researchers, practitioners, educators, and students. Our books will also expose the rich history of computing and the substantial global impact this field commands.

ACM Books will be published in both digital and print formats. The digital formats will include downloadable PDFs and reflowable ebook formats, such as EPUB, Kindle, and HTML. The print volumes will utilize the most cutting-edge, digital print-on-demand technology available.

For many future readers, the ACM Digital Library will be the primary channel from which to search, browse, and download complete copies or individual chapters of ACM Books. Libraries and library consortia worldwide will make buying decisions on behalf of their faculty and students; corporate libraries on behalf of their employees; and end users will have "unrestricted" and "essentially" free access to these books as a result. Pricing for institutions will be made affordable by ACM, in keeping with our long-standing tradition and non-profit approach to selling the ACM DL.

For professional and student members, the series will be available at a low "add-on" price to their current ACM Membership ($29 annually for professionals and $10 annually for students). For all other individuals interested in purchasing single print or digital copies, there will be an option to do so through leading booksellers, such as Amazon and B&N.com.

To accelerate our development of ACM Books, ACM is partnering with respected book publisher Morgan & Claypool. Many in the computing community will be familiar with the San Francisco-based M&C as a result of their highly successful series in computing called the "Synthesis Lectures," which publishes concise volumes across a range of computer science subjects.

There are two questions I am often asked when I talk to colleagues about the new series. The first relates to a book program ACM had in the 1990s called ACM Press, which was subsequently discontinued. Why, they ask, do we expect the new series to be a success when this earlier effort had failed? There are a number of answers to this question. Most important, as I noted, is the changing academic and publishing landscape. We are in a very different world now in terms of the number of players, publishing platforms, and related models for publication. Today, ACM is in a far better position to succeed in a digital world than we ever were in an analog world, which is in large part due to our positioning of the ACM DL, the increasing international profile of ACM's membership, and the revitalized opportunities in the scholarly book market.

The second question asked is how ACM Books is positioned with respect to M&C's Synthesis Lectures, NOW Publishers' Foundations & Trends series, or Springer Briefs. These publications have gained significant favor among the community in recent years. We are positioning ACM Books to complement these other products so that we enrich the computing publication ecosystem rather than constrain it. However, there are several important differences. Our primary objective is to serve our members and the computing community. This orientation will profoundly impact and determine our publishing program. Second, there is a difference in the nature of the books themselves, in length and breadth. These other publishers offer monographs or papers that range about 75–125 pages. ACM Books will present topics in far greater depth and detail. Third, and perhaps most importantly, there will be a difference in coverage. Rather than focus on narrow topics as these other series do, ACM Books will target topics more broadly and address areas not covered elsewhere.

We are currently in the process of forming the editorial board, establishing publishing guidelines and styles, and setting up a Web presence for the series. We are contacting ACM SIGSs for their input and their assistance in finding qualified area editors. By the time you read this, our website should be up and running at http://books.acm.org. I look forward to getting your input about any aspect of this program; please write to me at tamer.ozsu@uwaterloo.ca.

M. Tamer Özsu, a professor in the David R. Cheriton School of Computer Science at the University of Waterloo, is the editor-in-chief of ACM Books.

DOI:10.1145/2534706.2534708

Vinton G. Cerf

Software at Scale

I have been thinking about the implications of the increasing number of programmable devices and appliances that surround us. For one thing, they all require software to function. Who will write all that software? Of course, much of the software will simply be copies of the same code, so it is not as if we will need to write distinct software for each of the 50 billion devices. On the other hand, some of these mobile devices will be general-purpose platforms, such as those carried by many of the world's population. These support hundreds of thousands of special-purpose applications.

Thirty-five years ago, a similar kind of software proliferation was triggered by the development of VisiCalc by Dan Bricklin and Robert Frankston. This spreadsheet program transformed the Apple II desktop computer from a hobbyist platform to a business tool platform. Literally hundreds of thousands, if not millions, of people learned to write spreadsheet programs. One did not have to be a professional programmer or have a degree in computer science or electrical engineering to code a spreadsheet. Moreover, some of these spreadsheet programs were and are large and extremely complex. VisiCalc was overtaken by more powerful spreadsheet programming platforms but the principal point regarding the large population of spreadsheet programmers is still valid. Almost anyone could learn to write simple spreadsheets.

Mobile applications fall into a similar category although they typically require somewhat more sophistication even for relatively simple cases. Programming environments take a good deal of the complexity away. There are literally dozens of programming tools, software development kits, and mobile simulators available to aid in the development of a mobile application. The same may be said for Web-based programs that rely on browsers to interpret and execute application code or to interact with servers in the cloud. Without trivializing the effort required to program mobile, tablet, and laptop applications, it seems fair to observe that many of these programs have been and are being produced by self-taught programmers or at least those who may not have taken formal degrees in software-related disciplines.

Much of the software that underlies these conveniently programmed applications is deeply dependent on experienced, skilled, and often formally educated software developers. The so-called "warehouse computing" of the cloud relies on some of the most sophisticated operating system, application platform, and networking software ever developed. The operating systems supporting mobile devices, laptops, or desktops are similarly the products of sophisticated programming by extremely talented and experienced software developers.

I have written in the past about the notion of loosely coupled systems and the important role that standards play in their operation. The same principle is at work here. The underlying operating systems, software development kits, and networking facilities rely on standard protocols and application programming interfaces to hide complexity and allow programmers to focus on the application and not so much on all the infrastructure that must work for their applications to function. It has been observed that it is easy to write software that is hard to use and hard to write software that is easy to use. The same notion applies to the software development environments enabling a remarkable range of users to produce their own software. The most enabling software development environments represent considerable programming effort.

Interestingly, this line of reasoning leads me to believe that everyone would benefit from exposure to some form of programming if only to experience the kind of thinking that goes into making a programmable device do something. This is one reason ACM is so active in promoting computer science as a course equal in stature to physics, chemistry, biology, and mathematics in secondary schools. It should be permitted to fulfill STEM requirements in middle and high school curricula. The intent is not to turn everyone into professional programmers. Rather, it is to create familiarity with the concepts of programming so the software-based tools and platforms that surround us can be applied at need. Interested readers are referred to code.org for ideas along these lines.

I am reminded of a prediction made in the 1930s that by the 1950s, every citizen living in the U.S. would have to become a telephone operator to handle the rapidly growing use of the telephone. In one sense, the prediction was correct. The invention of direct distance dialing made all of us operators!

The increased use of programmable devices in daily living creates demand for software skills ranging from those needed for relatively simple, ad hoc applications to extremely sophisticated and large-scale systems design and implementation. Many of the latter will demand in-depth and formal training and perhaps even certification for applications involving risk to life and limb. ACM is right to pursue its strong advocacy for computer science in school curricula.

Vinton G. Cerf, ACM PRESIDENT

DOI:10.1145/2534706.2534709

Free the Digital Natives

James Geller raised important questions in his letter to the editor "Beware BYOD" (Sept. 2013) but mixed learning and assessment with the practicalities of supporting multiple devices. He highlighted possible outcomes—"distraction, cheating, more cheating, still more cheating"—along with the possibility that incompatibility between devices could result in poor grades, but failed to propose ways to address them.

In today's digital environment, where the simplest online search can turn up more information than will ever be found in a physical classroom, educators must confront the always-present interconnected nature of these devices by changing the way they teach. The traditional model of didactic lecturing from a podium before a classroom of students should complement collaborative activities, where students learn to research, filter, judge, and create their own learning. Educators must provide an "on-ramp" for them to acquire these skills in the context of the subject being covered and ensure assessment practices are an opportunity to demonstrate individual achievement.

Software and hardware fragmentation has always been an issue for educators, but with declining educational budgets in public institutions across Europe and the U.S., we must acknowledge that many students arrive with better technology in their pockets than is likely to be provided by the academy. Free/open-source software and virtualization are potential solutions, either standardizing on technologies students access and install themselves or providing applications not available through a virtualized server platform.

The related potential solutions are neither straightforward nor cheap, but educators must at least be willing to acquire the skills needed to address the changing worldview and technical competence of the students in their classrooms, who, as digital natives, will have always had access to these technologies.

Barry Avery, London, U.K.

Idempotence More Than Ever

Pat Helland's article "Idempotence Is Not a Medical Condition" (May 2012) addressed a serious reliability topic—messaging in a service-oriented world—but in an irreverent way. His axioms (such as "Every application is allowed to get bored and abandon its participation in the work.") are generally obvious when being read but often ignored by software developers in the real-world press of development.

Along with the proliferation of tools supporting distributed Web apps, it is easier than ever for software developers to (mostly) ignore the pitfalls of underlying distributed messaging, at least until the software is stressed in a production environment. Helland pointed us toward the dragons lurking in our assumptions concerning the robustness of messaging in a networked environment, concluding with "basic principles" cast as "four insidious illuminations":

- Because retries happen, all messages must be indempotent;
- Messages can be reordered;
- Hidden effects can cause one's dialogue partner to miss part(s) of a conversation; and
- Guaranteed delivery of the last message is impossible.

...ignorance of which can result in latent problems to surface only when repair is most costly.

Kudos to Helland for illuminating a significant source of bugs, amusing us in the process.

Steven Pothier, Tucson, AZ

Educators must confront the always-present interconnected nature of these devices by changing the way they teach.

Innovation Vs. Pride in Disruption

Moshe Y. Vardi's editorial "Has the Innovation Cup Run Dry?" (Sept. 2013) offered two divergent conceptions of innovation: In one, drawn from a 2013 Bard College commencement address by outgoing U.S. Federal Reserve chair Ben Bernanke (http://www.federalreserve.gov/newsevents/speech/bernanke20130518a.htm), innovations (1913–1963) were described as having produced "...dramatic improvement in the quality of daily life..." In the other, a report from the McKinsey Global Institute (http://www.mckinsey.com/insights/business_technology/disruptive_technologies) predicted (for 2013–2025) the emergence of "...technologies [having] the potential to disrupt the status quo, alter the way people live and work, and rearrange value pools."

A good way to view the divergence is to focus on what is changing in the first (quality of daily life) and the direction of that change (dramatic improvement). In the second, how people live and work is changing. The direction of that change (disrupt, alter, and rearrange) is far from unambiguously positive and likely to be viewed negatively by many.

I was thus prompted to explore ideas introduced by the influential Austrian-American economist Joseph Schumpeter (1883–1950), who used the term "creative destruction" in describing his theory of innovation-driven economic growth. In it, innovation operates within the system of production to cause old inventories, ideas, technologies, skills, and equipment to become obsolete. Replacement benefits consumers who experience improvement in daily life through increased capability, variety, and affordability of products and services. Part of the cost of this improvement is disruption in the world of work, as experienced by workers, managers, business owners, and investors, some benefiting and some forced to endure loss and painful adaptation. Schumpeter's theory thereby contributes to resolving the divergence of conceptions by showing how the effects of innovation once fell unevenly on separate aspects of life—outside work and within work.

Concerning those whose work contributes to innovation through technology, this analysis highlights challenges and their related questions, including:

Lost distinction. Why and how did the world of technology lose interest in the important distinction between effect on work and effect on daily life?;

Pride in disruption. Why and how did the world of technology begin to feel pride in disruption as such, usually expressed without comment, along with or in place of pride in improvement of daily life?;

Degraded communication. Have the loss of distinction and pride in disruption contributed to degraded communication between the world of technology and the world at large?;

Diminished effectiveness. Have degraded understanding and communication subtracted from the ability of the world of technology to satisfy human needs in the world at large?; and

Friction. Have degraded understanding and communication contributed to friction between the world of technology and the world at large?

Although I have not conducted an extensive review, I can say the world of technology gives them insufficient attention.

Robert E. Levine, Sierra Vista, AZ

More Fair Than Just Envy-Free

Although Ariel D. Procaccia's article "Cake Cutting: Not Just Child's Play" (July 2013) offered an interesting overview of recent research on cake division in computer science, it emphasized envy-free solutions that are not sufficiently fair.

Envy-freeness is a property that violates the axiom of monotony, or the requirement of Pareto-efficiency, in which a solution that improves the outcome of an agent by any amount is preferable in terms of fairness to a solution in which all agents have equal smaller outcomes. However, such a solution cannot be envy-free.

Envy-free solutions for distribution of multiple goods involve a subtler theoretical shortcoming. Consider allocating food to animals in a zoo. Though the animals have utilities that can be expressed in terms of the calories they consume, they can consume only certain foods. One might eat only eggs. Another might eat eggs but also other foods. A non-envy solution might therefore be to give the egg eater insufficient eggs for its survival. This theoretical weakness is alleviated through the concept of max-min fairness.

Why and how did the world of technology lose interest in the important distinction between effect on work and effect on daily life?

Procaccia mentioned utilitarian and egalitarian fairness but none of the important theoretical advances in the definition of distributional fairness. For example, Ogryczak et al.[2] formulated "equitable optimality," generalizing both utilitarian and egalitarian approaches by imposing three axioms on the preferences of fair solutions: impartiality (the permutation of outcomes from another solution should be indifferent); monotony; and the Pigou-Dalton principle of transfers (a small amount from a better-off agent to a worse-off agent should be preferred). It can be shown that finding equitably optimal solutions does not increase computational complexity by more than a polynomial factor. Such solutions can be found as Pareto-optimal solutions to a problem involving criteria obtained first from the Lorenz curve of the original distribution problem. Moreover, optimal solutions can be selected with consideration for the trade-off between equality and efficiency, increasing the likelihood of their acceptance by stakeholders.

The concept of envy-free solutions is useful only if agents are fully autonomous and selfish in a non-cooperative setting. However, Procaccia's example of CPU and RAM allocations to computing jobs does not require such a model. In most practical cases, there is a single administration of the computing resources, either cloud or grid, that can and should impose a better solution than the envy-free solution to the cake-cutting problem. It also illustrates that equitably optimal solutions requiring a cooperative setting can be obtained in practice.

A body of literature has aimed to find equitably optimal solutions in various settings, of which we recommend Luss[1] and Wierzbicki.[3]

Adam Wierzbicki, Warsaw, Poland
Włodzimierz Ogryczak, Warsaw, Poland

References
1. Luss, H. *Equitable Resource Allocation: Models, Algorithms and Applications.* Wiley, Hoboken, NJ, 2012.
2. Ogryczak, W. et al. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research 158*, 2 (Oct. 2004), 362–377.
3. Wierzbicki, A. *Trust and Fairness in Open, Distributed Systems.* Springer, Berlin, Heidelberg, 2010.

Author's Response:

I welcome these points and am happy to respond. First, many of the solutions I discussed in my article are both envy-free and Pareto-efficient. Second, in most fair-division settings envy-freeness implies proportional shares of the resources; in the zoo example, assuming eggs are allocated, the poor egg eaters would be envious. Third, I interpret the letter's penultimate paragraph primarily as a criticism of strategy-proofness rather than of envy-freeness. However, it is not the jobs that are autonomous but rather the users running them; the scheduler can impose a solution based only on the available information, which can be manipulated.

Ariel D. Procaccia, Pittsburgh, PA

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

The *Communications* Web site, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/2534706.2534710 http://cacm.acm.org/blogs/blog-cacm

The Lure of Live Coding; The Attraction of Small Data

Mark Guzdial ponders a new set of research questions, while Valerie Barr considers the utility of one person's data.

Mark Guzdial "Trip Report on Dagstuhl Seminar on Live Coding"
http://cacm.acm.org/blogs/blog-cacm/168153-trip-report-on-dagstuhl-seminar-on-live-coding/fulltext
September 26, 2013

I recently spent time at the remarkable Schloss-Dagstuhl in a seminar on "Collaboration and Learning through Live Coding," organized by Alan Blackwell, Julian Rohrhuber, and Alex McLean (and James Noble, who was not able to join us). It was a terrific event that has me thinking about a whole new set of research questions.

Live coding is a performance where programmers improvise music (and sometimes visuals) in front of an audience. It is real-time coding to create real-time music. Often, this occurs in pairs, trios, or even larger groups (see a quartet at http://www.youtube.com/watch?v=jlx6KNo5Eok). One programmer starts his music going, then the other one writes some code to respond and interact, and they both go off. It is a jam session on laptops.

The main live coding website is TOPLAP.org. There is another website devoted to AlgoRave—dance parties, driven by live coders generating the music. I still find that a little strange to think about: people dancing to some hacker's algorithms. For an introduction to live coding, I recommend this video from Andrew Brown (at http://vimeo.com/74081305), who wrote the JMusic library for music composition in Java that we use in our Media Computation curriculum, and also this video from Sam Aaron (at http://vimeo.com/22798433) on live coding in Overtone. Sam Aaron also created Sonic Pi, which is a synthesizer programming environment for the Raspberry Pi.

Most of the attendees were live coders, but there were a number of us others who helped explore the boundary disciplines for live coding. The seminar explored the ramifications and research potential of this activity.

Robert Biddle was there to lead discussions about the software engineering implications of live coding. On the one hand, live coding feels like the antithesis of software engineering, or as one attendee put it, "an ironic response to software engineering." There is no test-driven development, no commenting, no development of abstractions (at least, not while live coding), no exploration of user needs. On the other hand, live coding (not necessarily with music) can be an important part of exploring a space. One could imagine using live coding practices as part of a conversation with a user about needs and how the programmer understands those needs.

Geoff Cox led a conversation about the humanities research directions in live coding. Geoff has a particular interest in labor, and he pointed out how live coding surfaces hidden aspects of the labor in modern society. While computing technology has become ubiquitous in the developed world, few people in our society have ever seen source code or a programmer. What does it mean for an audience to see an embodiment of a programmer, to see the labor of generating code? What is more, the audience is seeing code doing something that is not normally associated with people's notions of what code is for—making music. How does this change the audience's relation to the computing technology? The notion of programming-as-performance is an interesting and novel way of thinking about computing practice, and in sharp contrast to stereotypical perspectives of programming.

Thomas Green, Alan Blackwell, and others from the PPIG (Psychology of Programming Interest Group, http://www.ppig.org) community pointed to the notations that the live coders used. I have drawn on Thomas' work on the cognitive dimensions of programming and the implications of our programming representations since I was a graduate student. The language constraints for live coding are amazing. Live coders tend not to use traditional development languages like C++ or Java, but instead work in Scheme, Haskell, and a variety of domain-specific languages (like SuperCollider)—often building their own implementations. Live coders need languages that are expressive, provide for the ability to use techniques like temporal recursion, are concise, and (as one live coder put it) "let me live code at 2 A.M. at a dance club after a couple of beers."

I was there to connect live coding to computing education. I learned the connections from the seminar—I hadn't really seen them before I got there. I am fascinated by the humanities questions about introducing source code and programmers to a technologically sophisticated audience that probably never saw the development of code. I am also interested in the value of rapid feedback (through music) in the performance. Does that help students understand the relationship between the code and the execution? How does the collaboration in live coding (for example, writing music based on other live coders' music) change the perception of the asocial nature of programming?

Live coding is rich with interesting research issues because it is exploring such a different space than much of what we do in computer science. It is about expression, not engineering. It is about liveness, not planfulness. It is about immediate creation of an experience, not later production of an artifact. That makes it worth exploring.

Valerie Barr "The Frontier of Small Data"

http://cacm.acm.org/blogs/blog-cacm/168268-the-frontier-of-small-data/fulltext

September 29, 2013

I had the opportunity recently to attend a talk by Deborah Estrin, a professor of computer science at Cornell Tech in New York City, entitled "small, n = me, data." Her title is a play on our usual way of referring to problem size; in this case, making the "n" of the problem size just a single person. Certainly Estrin is not turning her back on big data and all that can be learned from it, but she is very interested in how a single person's data can be used to understand their situtation. For example, in the medical realm, the Open mHealth effort is developing an architecture that can integrate data from an individual's use of specific apps in order to help a health care provider make recommendations.

Particularly interesting is the small data effort that Estrin has undertaken at Cornell NYC Tech. Her definition of small data is "your row of their data." What observations would surface if you could analyze in some combined way your mobile usage, cable usage, utility usage, e-commerce activities, search activities, social media and email usage, automobile usage gathered from smart car data, use of games, music, and video? What changes in the health and well-being of an aging parent might surface through analysis of their aggregated data? Would one be better able to compare the efficacy of different courses of medical treatment by looking at aggregated data? For example, data traces showing how far you walked and how early in the morning you left the house could indicate relative effectiveness of an arthritis medication.

Estrin headed off possible issues with the comment "where there is a privacy concern, there is an opportunity." Her goal is to develop an "ecosystem of applications" that an individual can run over her or his own set of data streams, a collection she referred to as our "personal data vault." Her hope is that eventually we will be able to subscribe to our own individual data traces. Personal data APIs will allow for development of real-time personal data apps.

Estrin listed several key challenges:

1. getting the data
2. processing and making sense of noisy, diverse data
3. secure models for the personal data vaults
4. a testbed for app prototypes

You can read Estrin's 2013 TEDMED talk online (at http://smalldata.tech.cornell.edu/narrative.php) to get more information. She closed the talk I attended with a reminder that Cornell NYC Tech is actively recruiting graduate students, so pass along the link to their Admissions page (http://tech.cornell.edu/admissions/). I know I definitely have students who will be very interested in the possibility of working on this small data project. ■

Mark Guzdial is a professor at the Georgia Institute of Technology. **Valerie Barr** is a professor at Union College.

Association for
Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD13

You can join ACM in several easy ways:

Online	Phone	Fax
http://www.acm.org/join	*+1-800-342-6626 (US & Canada)* *+1-212-626-0500 (Global)*	*+1-212-944-1318*

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:
http://www.acm.org/membership/L2-3/

Special rates for members of sister societies:
http://www.acm.org/membership/dues.html

Please print clearly

Name

Address

City | State/Province | Postal code/Zip

Country | E-mail address

Area code & Daytime phone | Fax | Member number, if applicable

Purposes of ACM

ACM is dedicated to:
1) advancing the art, science, engineering, and application of information technology
2) fostering the open interchange of information to serve both professionals and the public
3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/about/code-of-ethics

choose one membership option:

PROFESSIONAL MEMBERSHIP:

o **ACM Professional Membership: $99 USD**

o **ACM Professional Membership plus the ACM Digital Library: $198 USD ($99 dues + $99 DL)**

o **ACM Digital Library: $99 USD (must be an ACM member)**

STUDENT MEMBERSHIP:

o **ACM Student Membership: $19 USD**

o **ACM Student Membership plus the ACM Digital Library: $42 USD**

o **ACM Student Membership PLUS Print *CACM* Magazine: $42 USD**

o **ACM Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 USD**

**All new professional members will receive an ACM membership card.
For more information, please visit us at www.acm.org**

Professional membership dues include $40 toward a subscription to *Communications of the ACM*. Student membership dues include $15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:
Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

o Visa/MasterCard o American Express o Check/money order

o Professional Member Dues ($99 or $198)	$ ________
o ACM Digital Library ($99)	$ ________
o Student Member Dues ($19, $42, or $62)	$ ________
Total Amount Due	$ ________

Card # | Expiration date

Signature

Science | DOI:10.1145/2534706.2534711 Chris Edwards

Life Points the Way to a New Template for Chipmaking

While significant obstacles remain, researchers are optimistic about using DNA to guide graphene into complex circuit shapes on silicon.

DNA, THE BUILDING block for our genes, may provide the key to the bottom-up assembly of complex, nanometer-scale circuits, following research performed by scientists based at Harvard University and the Massachusetts Institute of Technology (MIT) that used the biomolecule to carve patterns in graphene lying on the surface of a silicon wafer.

However, researchers need to overcome significant obstacles to make the processes viable on an industrial scale.

Today, integrated circuit (IC) manufacturers use exclusively top-down techniques to define the transistors and wiring that make up ICs based on optical lithography. Light shone through a reticle defines features on a chemical resist spun onto the surface of a silicon wafer. The resist typically hardens where the light strikes but remains soft enough to wash off elsewhere, so that materials can be deposited or implanted in the areas not masked by the resist.

Almost 20 years ago, the semiconductor industry moved to use light in

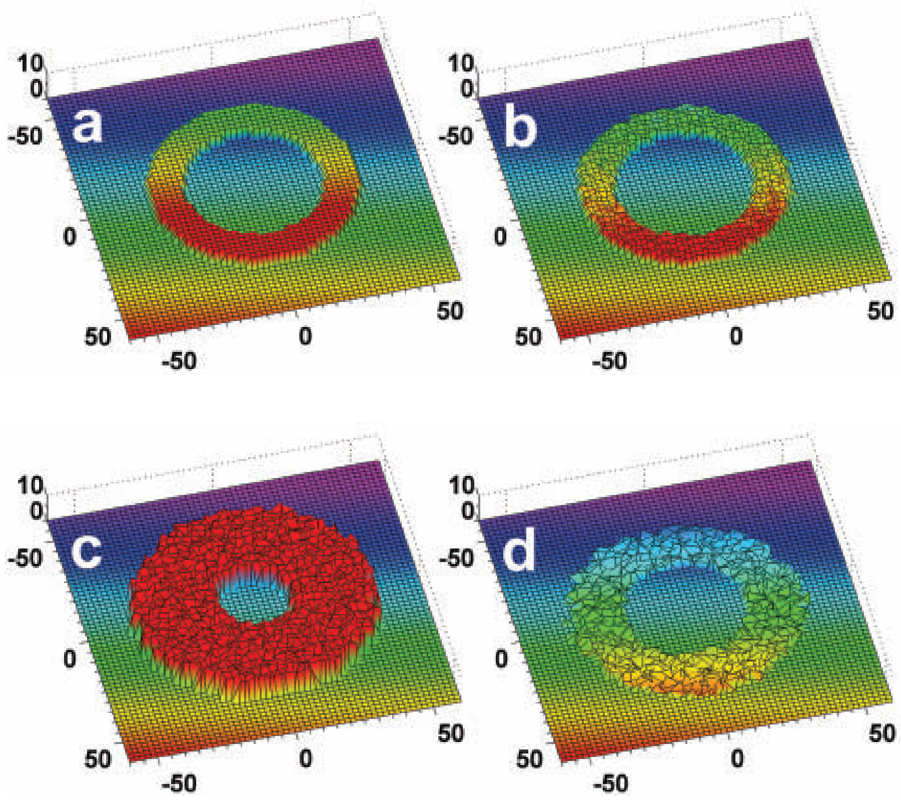


IMAGE BY ZHONG JIN

In images a through c, metallized DNA (in red) is built up to form the letter O on a graphene substrate. In image d, treatment with oxygen plasma has etched the letter into the graphene.

the sub-250nm deep ultraviolet spectrum to try to keep pace with shrinking feature sizes, which reduced to 250nm only a few years later. This led to increasing problems with optical factors such as diffraction. The impact of blurring caused by diffraction increases dramatically as feature sizes approach the wavelength of the light that bends around them.

Since then, because of problems in sourcing compatible materials for much smaller wavelengths, chipmakers have been forced to stick with deep ultraviolet light, now at 193nm. However, the minimum feature size has been reduced tenfold, to less than 25nm. Even in the early 2000s, when the differential was far lower, Dan Hutcheson, analyst and CEO at VLSI Research, likened the practice to "painting a one-inch line using a four-inch brush."

Chipmakers fought back through the use of increasingly complex masks that contain billions of tiny dummy features. These dummy features introduce constructive and destructive interference that result in the intended circuitry being exposed on the surface of the wafer. Now, however, they are at their limit.

The industry hopes to move to extreme ultraviolet (EUV) light, using a sub-15nm wavelength, but the machines will be expensive and still cannot rival the throughput of today's deep-ultraviolet equipment. So, the industry has employed a stopgap called double patterning in which different, complementary masks slightly offset from each other expose the wafer in two steps to form a final image. A variant of this process—self-aligned double patterning—points the way toward bottom-up self-assembly.

With the self-aligned version of double patterning, a second resist is coated over the hardened resist left after exposure. Chemical etching that follows eats away all of the resist except for the second resist that lies at the edges of the original, creating two thin lines from the single original. A version of this technique is already used to form the fins of Intel's recently introduced trigate transistor, and Intel's Director of Advanced Lithography Yan Borodovsky says repeated chemical processing can perform what he calls "pitch division" many times over. Applied Materials, which supplies resists and other chemicals to the chipmakers, has developed techniques using block copolymers that use similar chemical processes to create more complex, regular features on the surface of a wafer that are templated by the materials left by optical lithography.

According to Lars Liebmann, distinguished engineer at IBM's Semiconductor R&D Center, self-assembly using block copolymers can yield sub-10nm resolution features in processes that work with today's lithography equipment. However, the impact on design is severe, he says, forcing designers to use almost entirely regular structures.

DNA may provide a way of making self-assembled structures less regular and ease the job of the chip designer. Michael Strano, based at MIT and a member of the team that used DNA to produce patterned graphene, says: "You cannot program block copolymers the way you can with DNA. Copolymers can yield simple shapes and lines, but you cannot dial in anything else. They cannot make a crossbar or anything like that."

In the early 1980s, Ned Seeman at the State University of New York demonstrated that strands of DNA could be designed to self-assemble into 3D shapes that could be linked together in large regular lattices. In the mid-2000s, Paul Rothemund of the California Institute of Technology further developed the technique to create what he called "scaffolded DNA origami."

DNA is made up of sequences of four types of monomer or 'bases' that combine into a single-stranded polymer held together by a phosphate backbone. Each monomer binds to a complement through hydrogen bonds. These bonds use the electrostatic attraction between electrons on atoms such as oxygen, and the proton that is the hydrogen nucleus. The difference between the structure of the two pairs of bases means they preferentially bind to each other. Adenine forms two hydrogen bonds with thymine; guanine forms three bonds with cytosine. Further electrostatic interactions help form DNA into its familiar double-helix configuration.

"You cannot program block copolymers the way you can with DNA. Copolymers can yield simple shapes and lines, but you cannot dial in anything else."

DNA origami works by artificially producing sequences that are not entirely complementary and so do not simply wrap into a 2nm-wide helical ribbon. Rothemund's original technique combined a scaffold of single-stranded DNA 7,000 bases long with a set of much shorter, 'staple' sequences, each designed to bind to two different parts of the scaffold. These staples caused the scaffold to form a series of hairpin loops that, by careful design of the sequences, would form arbitrary shapes such as triangles, stars, letters, and smiley faces. Since then, researchers have used other approaches to increase the complexity and size of the shapes they can produce.

A further possible advantage of using DNA as a programmable structural material is that it can be used to perform computations as it assembles. Researchers like Erik Winfree, based at the California Institute of Technology, have used reactions that sever, shuffle, and displace strands of DNA to perform calculations.

Peng Yin, assistant professor in the Department of Systems Biology at Harvard University, who collaborated with Strano's team at MIT, said, "Computational assembly provides a fascinating direction to take. We could use rules to assemble blocks. One potential advantage is that you could build very sophisticated structures using a small number of building blocks.

"People get excited about using DNA for pure computation that may not be fruitful because other computers, such as those based on silicon, are already very powerful. I do not expect banks of DNA molecules to compete with an iPhone, but when DNA performs computation, you are integrating computation with molecular physical behavior, instead of just using self-assembly."

Rothemund says there is still a lot of development to be done before DNA can be relied upon to build electronic structures algorithmically. "It has been shown theoretically that algorithmic self-assembly can grow patterns that would define demultiplexer circuits, for example. This requires the assembly to do a simple computation: counting in binary as it grows. Experimentally, binary counters have been achieved, but their error rates are still quite large; we cannot get them to higher than a few dozen routinely. When tiles attach incorrectly, the counters can jump to other numbers."

Although DNA has apparent advantages over simple block copolymers, it has significant drawbacks. A strongly hydrophilic material, it does not adsorb readily on some of the materials that may be used for ICs in the future, such as graphene. Also, it cannot survive a number of the processes currently used in industrial semiconductor manufacturing, such as acid and dry-plasma etching.

The Harvard-MIT team solved the problem of adsorbing DNA onto graphene by depositing a layer of another organic material on the surface, to which the DNA more readily attaches. So that the DNA template could act as a mask for the plasma-etching process used to remove unwanted graphene, the group used a technique derived from one developed at Duke University in 2003 to coat the DNA with gold. A downside of the metallization process is that it has a rougher edge than the original DNA, says Strano.

Rothemund says, "Many of the most interesting properties of graphene depend exquisitely on the exact conformation of its edges. Pattern transfer from DNA origami is not high-resolution enough to define these edges, so perhaps some chemical post-process can heal them or help ensure they do not have defects which will hurt their electronic properties."

An alternative approach to protecting the DNA template may work with more conventional devices. Yin has collaborated with a team based at the University of Pittsburgh to develop techniques to deposit silicon dioxide —a material used to form hardened resists for chipmaking—either around the DNA or in place of it. "The process lets us transfer the shapes of DNA to other inorganic materials," says Yin, which could include the conventional silicon-based materials used in today's chipmaking fabs.

"When DNA performs computation, you are integrating computation with molecular physical behavior, instead of just using self-assembly."

Rothemund says a more viable application may be to use DNA to help attach different materials and nanostructures, such as quantum dots and carbon nanotubes, to ICs. "Conventional lithography has difficulty organizing a lot of these materials," he says. "In particular, I am excited about metallic nanoparticles because of their optical properties. Researchers have spent years characterizing the unusual optical spectra of clusters of nanoparticles, which were 'assembled' by chance. DNA origami has the ability to not only create clusters of nanoparticles, but also to arrange them in regular grids."

Other biomolecules such as proteins—which are themselves built using the genetic information contained in natural DNA—could be used as programmable polymers that are more resistant to industrial processes, or which bind more easily to chip surfaces.

Says Yin, "We expect the same self-assembly principles could be extended to new materials. The most fundamental thing is that DNA is a digital polymer. You can encode a digital sequence with very specific base pairing, but it is not restricted to DNA; it can be extended to other molecules."

Strano says DNA currently has an important advantage: "We have not solved the protein-folding problem; their shapes are very difficult to predict computationally. But we have figured out the DNA folding problem. DNA folds in a very unambiguous way, so much that if you have DNA sequences floating around in solution, they will eventually find their complement and fold."

Similar binding dynamics have been applied to chemically modified forms of DNA and other molecules, such as peptide nucleic acid (PNA), which combines a more robust protein backbone with the complementary bases of DNA. Other significant issues still need to be fixed before programmable digital polymers become industrially useful in building.

"Conventional lithography allows arbitrary patterning over enormous length scales, whereas the largest arbitrarily patterned DNA nanostructures achieved so far are roughly a micron in size. And conventional lithography achieves defect rates which will be difficult to achieve with these self-assembled structures," Rothemund says.

He adds, "There are some new DNA nanostructures which, with great effort to drive down defect rates, may yield masks which can compete with block copolymers, if directed self-assembly of block copolymers is ever considered practical." C

Further Reading

Rothemund, P.W.K.
Folding DNA to create nanoscale shapes and patterns, *Nature 440*, 297, Mar. 16, 2006

Zhang, G., Surwade, S.P., Zhou, F., Liu H.
DNA nanostructure meets nanofabrication Chemical Society Reviews 2013, 42, 2488

Jin, Z., Sun W., Ke Y., Shih C-J., Paulus, G.L.C., Wang Q.H., Mu B., Yin, P., Strano, M.S.
Metallized DNA nanolithography for encoding and transferring spatial information for graphene patterning *Nature Communications* 4, 1663, Apr. 9, 2013

Surwade, S.P., Zhou, F., Wei, B., Sun, W., Powell, A., O'Donnell, C., Yin, P., Liu, H.
Nanoscale growth and patterning of inorganic oxides using DNA nanostructure templates *Journal of the American Chemical Society*, 135 (18):6778-6781, May 8, 2013

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

Technology | DOI:10.1145/2534706.2534712
Samuel Greengard

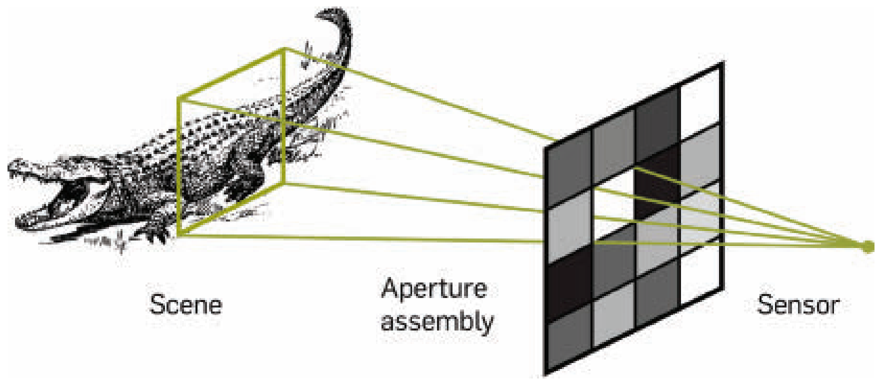
Seeing the Big Picture

Lensless cameras and other advances in digital imaging, computational optics, signal processing, and big data are transforming how we think about photography.

OVER THE LAST decade, digital photography has revolutionized the way people snap, store, and share photos. It has unleashed remarkable features and capabilities that have transformed the way we think about images…and the world around us. Yet, for all the advances in megapixels and signal processing, one fact remains: "Today's digital cameras were designed to replace analog information-gathering devices such as film," observes Richard Baraniuk, a professor of electrical and computer engineering at Rice University.

However, times and technology are changing. Researchers are exploring how to use computational optics and digital imagery in new and innovative ways. Lensless cameras, single-pixel imagery, devices that can see around corners and a number of other technology breakthroughs could transform photography and computational optics in ways that would have been unimaginable only a few years ago. These devices—particularly as they capture images outside the visible light spectrum—could address a wide array of challenges, from managing traffic networks and fighting crime to improving medicine.

That is no small matter, particularly as networks expand and mobile technology matures. Says Paul Wilford, senior director for Bell Labs, which has designed and built a prototype lensless camera: "As we move into a world where we rely on cameras and sensors to monitor and manage an array of tasks, we will need new types of devices and technological breakthroughs. We must find new ways to capture, compress, and use imagery."



At top, a diagram of how Bell Labs' lensless camera works: its single-pixel sensor is arrayed behind an aperture assembly that can create a matrix of apertures of varying opacity, so multiple measurements of light data can be conducted at once. Bottom, the lensless camera (black box) undergoes testing.

Pixel Perfect

Venturing into the future of digital imagery requires researchers to fundamentally rethink—but also revisit—the concept of photography. The traditional method of acquiring an image requires a lens and some type of medium (typically light-sensitive photographic paper or a sensor) to capture photons as dots or pixels. These techniques evolved from the concept of a *camera obscura*, a device that dates back thousands of years, which uses a pinhole to capture light and display images. Among its first uses were the viewing of solar eclipses and the patterns of shadows cast by trees and other objects.

Modern digital photography is incredibly effective at capturing images. Today's cameras store data in millions of pixels and rely on sophisticated sensors, signal processing, and algorithms to produce extraordinarily high-quality photographs. Over the last decade, conventional film has largely disappeared and digital cameras built into smart devices are now ubiquitous.

The goal of today's researchers is not to replace these cameras. Yaron Bromberg, a post-doctoral associate at Yale University, believes capturing images from non-visible parts of the spectrum, including infrared and terahertz radiation, could lead to breakthroughs in fields as diverse as medicine and environmental research. He is in a group that has studied pseudothermal compressive ghost imaging and how to build a single-pixel light capture system. "Today, devices that capture images in non-visible parts of the spectrum are expensive. Future cameras could open up a vast array of possibilities."

Others, including Wilford, share that vision. "There have been fantastic advances in digital cameras, particularly in optical wavelengths. We can capture millions of pixels at a rate of hundreds of times per second with remarkable resolution. We also have coding algorithms that reduce file size and bandwidth," notes Wilford. Capturing images in new ways and designing new techniques to compress files remains at the frontier of computational optics.

The Bell Labs team has developed a so-called "lensless camera" that relies on compressive sensing techniques to capture images. The system, built from off-the-shelf components, captures light via a two-dimensional array of photoelectric sensing elements built into an LCD panel. A micromirror array performs the functions of both pixelization and projection (attenuating aperture layers are used to create a pinhole which forms an image of the scene on the sensor array). Each array randomly records light from the scene; a sensor is able to detect three colors. Researchers can control each aperture element independently in accordance with the values of the overall sensing matrix. By analyzing multiple images, extrapolating on the data and redundancies and then correlating data points, it is possible to create a photographic image.

Bell Labs engineer Gang Huang says the accuracy of the lensless system—which can capture both visible and non-visible light—improves with more snapshots of the same object. However, the goal is not to obtain an attractive high-resolution image suitable for online viewing or a photographic print. "The appeal is that we are able to obtain a good image with only a fraction of the data that is required for a conventional image," he says. The system can reduce file size by factors of anywhere from 10 to 1,000, all while eliminating the need to focus a lens. In fact, the virtual image is not formed by any physical mechanism and, in the end, no planar image is ever created. The quality of the image is affected only the quality of the resolution.

Lensless cameras and other technology breakthroughs could transform photography and computational optics in ways unimaginable only a few years ago.

The technology creates new possibilities and opportunities. For example, it could prove valuable for monitoring traffic across a network of roads or highways. "If you have 1,000 cameras on the road and each produces a single pixel, the resulting pixel stream requires much less data," Wilford explains. The lensless system is able to detect infrared and millimeter waves; this could make it valuable for monitoring airports, stadiums, businesses, and other facilities. Alterations in consecutive images could alert a security team that something has changed, and indicate the speed of the change.

Although the Bell Labs group has successfully tested the lensless camera, the obstacle for now is the size of the device, which measures about a foot in height, width, and depth, notes Hong Jiang, a mathematician and signal processing expert for the team. Over the next few years, the researchers hope to dramatically reduce the size of the device while making it operate faster. "There are a lot of interesting possibilities and we have just begun to examine them," Wilford says.

Mixed Signals

Other researchers are also hoping to rethink and reinvent photography. At Rice University, Baraniuk has also explored the use of single-pixel image capture, a technique he pioneered with Rice University electrical engineering professor Kevin Kelly. His research has focused on using bacterium-sized micro-mirrors that reflect light onto a

ACM Member News

DRAY STUDIES INTERSECTION OF PEOPLE, TECHNOLOGY

When Susan M. Dray, president of Dray & Associates, Inc., received her doctorate in psychology from the University of California, Los Angeles, in 1980, there was "no such thing" as human-computer interaction (HCI), she recalls; computers were relegated to the back office.

Since then, Dray, a self-described "professional technology empath," has promoted the concepts of Human-Centered Design (HCD) and User Experience (UX). "I study the way people live and work to adapt technology to human needs," she says. To design a successful product, "you must comprehend user needs and the fabric of people's lives so the device or application fits their lives and fulfills their needs."

A founding member of ACM's Special Interest Group in Computer Human Interaction (which awarded her the SIGCHI Lifetime Service Award in 2005), Dray runs computer-engineering workshops and conducts research to assist developers designing applications and games for computers, smartphones, and other mobile devices.

Dray recently went to Nairobi to conduct a three-day course for 70 people, splitting them into teams to work on projects like mobile health applications. In March 2014, she will travel to Panama on a Fulbright Scholarship to train teachers in HCI at the Technological University of Panama.

Meanwhile, Dray actively pursues her other passion: mentoring. "I have 15 mentees. They take my ideas and kick them up several levels; it's exciting. I want to encourage the next generation and leave the world a better place," she says.

—*Laura DiDio*

single sensor; each mirror corresponds to a particular pixel. Just as high-resolution photographic images can be compressed by stripping out unneeded data and storing the file in JPEG format, it is possible to extrapolate on existing data in the image. The technique cycles through 50,000 measurements in order to find the optimal directional orientation for the mirrors.

The concept is appealing because a single-pixel design reduces the required size, complexity, and cost of a photon detector array down to a single unit. It also makes it possible to use exotic detectors that would not work in conventional digital cameras. For example, it could include a photomultiplier tube or an avalanche photodiode for low-light imaging, layers of photodiodes sensitive to different light wavelengths for multimodal sensing, or a spectrometer for hyperspectral imaging. In addition, a single-pixel design collects considerably more light in each measurement than a pixel array, which significantly reduces distortion from sensor nonidealities like dark noise and read-out noise.

These multiplexing capabilities, combined with shutterless sensing and compressive sensing, could lead to new types of cameras that might, for example, allow motorists to see through fog or to better see objects on the road at night, and thus avoid crashing into them or driving over them. Ultimately, the concept redraws the boundaries of imaging. "What we have come to realize," Baraniuk says, "is that digital measurements that we generate at a scene or in the world around us do not need to be exactly analogous with how we use a film or conventional digital camera."

There also is growing interest in engineering cameras that can see through solid objects. Researchers at Duke University, for example, have developed a camera that detects and records microwave signals. The device uses a one-dimensional aperture constructed from copper-based metamaterial to capture data that it sends to a computer, which constructs an actual image. Among other things, the device could prove valuable to law enforcement agencies; for example, passengers at airports could simply walk past the device at a security checkpoint while it scans for weapons and explosives. There would no longer be a need for long security queues.

"We are on the verge of remarkable breakthroughs in the way we think about photography and use computational optics."

Scientists are peering into other concepts that are equally mind-bending. For example, MIT associate professor Ramesh Raskar and a group of researchers are exploring the emerging field of Femto-photography; their goal is to build a camera that can see around corners and peer beyond the line of sight. Such a device could provide access to dangerous and inaccessible locations including mines, contaminated sites, and inside certain machinery. He describes the method as using "echoes of light" to capture information about the overall environment.

The concept, which the group has already tested successfully, uses a laser light burst directed at a wall or object. The camera records images every 2 picoseconds, the time it takes light to travel 0.6 millimeters. By measuring the time it takes for scattered photons to reach a camera—and repeating the process over 50 femtoseconds (50 quadrillionths of a second), it is possible to construct a view of the hidden scene. The system relies on a sophisticated algorithm to decode photon patterns. At present, the entire process takes several minutes, though researchers hope to reduce the imaging time to less than 10 seconds.

Baraniuk believes researchers will overcome many existing hurdles—particularly surrounding signal processing and computational analysis—over the next decade. They have already taken a giant step in that direction by constructing algorithms that sidestep conventional signal processing and instead mine big data. Over the next few years, as imaging systems and computers advance, once-abstract and seemingly unachievable photographic methods will become reality. One company, InView, has already begun to introduce cameras that use advanced imaging and compressive sensing techniques.

Werner says computational imagery increasingly will tie in augmented reality. He believes computational imaging and sensing will also meld with 3D cameras, night vision, adaptive resolution technology, holographic displays, and negative index material refraction. As hardware gets "faster and cheaper and algorithms become more sophisticated," next-generation cameras and image-capture devices will also connect with social networks and provide new ways for people to view and share the surrounding environment.

"We are on the verge of remarkable breakthroughs in the way we think about photography and use computational optics," Baraniuk concludes. In fact, "At some point, the end result might not be an actual image. The camera might make inferences and decisions based on certain parameters, including where a person is located in a room or the overall pattern of cars on a network of roads. We are moving into a world where there will be lots of different ways to sense what is taking place around us." **C**

Further Reading

Huang, G., Jiang, H., Matthews, K., Wilford, P.
Lensless Imaging by Compressive Sensing, IEEE International Conference on Image Processing, ICIP 2013, Paper #2393, May 2013. http://arxiv.org/abs/1305.7181

Katz, O., Bromberg, Y., Silberberg, Y.
Compressive Ghost Imaging, Dept. of Physics of Complex Systems, The Weizmann Institute of Science, Rehovot, Israel. http://arxiv.org/pdf/0905.0321v2.pdf

Baraniuk, R.G.
More is Less: Signal Processing and the Data Deluge, Science, Data Collections Booklet, 2011. http://www.ncbi.nlm.nih.gov/pubmed/21311012

Velten, A., Willwacher, T., Gupta, O., Veeraraghavan, A., Bawendi, M.G., Raskar, R.
Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging, Nature Communications 3, Article number: 745, Published 20 March 2012. http://www.nature.com/ncomms/journal/v3/n3/full/ncomms1747.html

Samuel Greengard is an author and journalist based in West Linn, OR.

Society | DOI:10.1145/2534706.2534713 Karen A. Frenkel

CS Enrollments Rise… at the Expense of the Humanities?

A growing proportion of U.S. college students are earning degrees in computer and information sciences, surprising some in academia.

It's common these days to hear computer science majors and young engineers at startups and large companies alike echo Steve Jobs, with talk of changing the world. The romance of innovation and the ability to give consumers technological products they want before they know it, as well as the ability to expand one's knowledge base while attaining in-demand job skills, are just some of the reasons the number of students seeking bachelor's degrees in computer science is rising—apparently at the expense of the humanities.

For centuries, proponents of the humanities have argued that these disciplines that study human culture, which include literature, philosophy, the classics, film studies, art history, music, and religious studies, provide the tools to reimagine and transform the world. Now, suddenly, they find themselves having to make that case anew.

A Harvard University report, "Mapping the Future," published in May, observed, "Just as the engineer makes life-transforming models through drawing on her *ingenium*, or imagination, so too the artist, and those emboldened to evaluation through responsiveness to art, imagine the remaking of an always recalcitrant world." "Mapping the Future" was written in response to falling numbers of bachelor's degrees granted in the humanities, both nationally and at Harvard.

The percentage of humanities bachelor's degrees awarded in U.S. institutions of higher learning slipped from 17% of all bachelor's degrees awarded in 1966 (88,503) to 7% (115,627) in 2011, according to figures published in 2013 from humanities indicators of the American Academy of Arts and Sciences (AAAS). (Although absolute numbers increased, so did the total number of bachelor's degrees awarded; consequently, the humanities' share of all bachelor's degrees remains well below the 1971 peak of 16% (136,213). The total number of bachelor's degrees conferred in 1970–1971 was 836,730, and in 2009–2010 the total was 1,650,014, according to the Digest of Education Statistics published by the National Center for Education Statistics.)

The AAAS also compared humanities degrees awarded to those granted in selected other academic fields, including engineering. In 1987, humanities bachelor's degrees awarded were 10% of the total bachelor's degrees granted, whereas engineering bachelor's degrees made up 11.4% of the total. In 2010, humanities degrees awarded made up 11.5% of the total bachelor's degrees granted, compared to 6.7% for engineering (AAAS did not offer absolute numbers). At Harvard, the percentage of students concentrating their scholastic efforts in the humanities has fallen from 36% in 1954 to 20% this year, based on the "Mapping the Future" report.

According to the National Center for Education Statistics, the number of bachelor's degrees in computer and information sciences and support services that were conferred in the U.S. rose 9% from 2010 to 2011, to a total of 43,072. The 2011–2012 Taulbee Survey, which tracks newly declared computer science majors in doctoral-granting programs, found a 29% jump, from 13,227 bachelor's degrees awarded at the end of the previous school year to 17,226 in 2011–2012, a "remarkable increase," according to Stuart Zweben, past president of ACM and professor emeritus of Computer Science and Engineering at Ohio State University, who co-authored the 2011–2012 Taulbee Survey. "This is the kind of one-year jump that's really hard to sustain," he says.

The Taulbee Survey found the total enrollment in bachelor's degree-granting computer-related programs in the U.S. (new computing students plus returning majors in computer engineering departments, information departments, and Canadian departments as well as U.S. computer science departments) was 67,850 in 2012, up from 60,636 in 2011.

Over the past decade, the peak for CS and IT education was 60,000 bachelor's degrees granted in 2004, according to the National Center for Education Statistics. It took a while after the 2000 dot-com bust for the pipeline to run its course and the number of relevant degrees to diminish; the nadir was 38,000 in 2009, as undergrads abandoned computer science in favor of other disciplines. However, between 2009 and 2010, the decline in the number of IT bachelor's degrees turned around, jumping 4%.

It is not clear how many humanities degrees are being lost directly to computer science and IT, but it is known that degrees do not necessarily translate into jobs related to workers' majors. The Harvard study noted an "intellectual diaspora" for humanities majors, and said the humanities still "represent solid launching pads into professional schools" in fields like law and medicine, "which admit humanities majors at similar, and sometimes higher, rates as any other branch of study."

Zweben says the uptick in computer science degrees is indicative of the impact of the country's push toward STEM technologies, from which CS benefits. Also, students see computing used pervasively—in hospitals, cars, movies, getting the news, and accessing social media. "It is all over the place, and the generation going to college today grew up with technology, so they are very comfortable with it and see what it can do," says Zweben.

Last spring, Hal Salzman, professor of Public Policy at Rutgers University, and colleagues Daniel Kuehn and B. Lindsay Lowell, interpreted National Center for Education Statistics data and what it means for the IT workforce. In the April 2013 paper "Guest Workers in the High-Skill U.S. Labor Market," Salzman et al. analyzed supply, employment, and wage trends, finding that there are multiple routes to IT employment, most of which do not require a STEM degree. "Only about a third of the IT workforce has an IT-related college degree. Thirty-six percent of IT workers do not hold a college degree at all. Only 24% of IT workers have a four-year computer science or math degree," the report said.

Contrary to many industry claims, the Salzman paper found U.S. colleges and universities provide an ample supply of highly qualified STEM graduates. Guestworkers may be filling as many as half of all new IT jobs each year, according to the report, and IT workers earn the same salaries today that they did 14 years ago.

IT recruitment site Dice.com in May also released a study, "America's Tech Talent Crunch 2013," which also cites National Center for Education Statistics. Besides the growth in CS and IT bachelor's degrees, the Dice study notes that associate's degrees in those fields rose 16% between 2010 and 2011 (totaling 37,677 in 2011), and increased 36% (from 28,000) over the previous four years. Between 2010 and 2011, 19 states conferred more two-year degrees than bachelor's degrees, the report says.

Asked what sorts of jobs IT workers with AAs can get, Dice.com chairman, president, and CEO Scot Melland said his site typically sees people with that level of qualification becoming entry-level software developers, helpdesk employees, and IT support personnel. "At some community colleges, students already have BAs in something else and come back for AAs in IT," says Melland. "We find them in project management or administrative jobs."

The Dice study says the rise in BAs and AAs with CS or IT degrees will crowd the field. "As the growing demand for tech workers meets a growing supply—a higher number of new two-year and four-year graduates entering the workforce—the result may well be more competitive pressure for job applicants and a tougher fight among the best and brightest for coveted jobs on

Milestones

Computer Science Honors

IACR ANNOUNCES NEWEST FELLOWS

The International Association for Cryptologic Research (IACR), a non-profit supporting the promotion of the science of cryptology. recently announced these additions to the ranks of the organization's Fellows:

▸ Dan Boneh, recognized for opening new areas in cryptography and computer security, and for innovative educational initiatives.

▸ Ronald Cramer, for contributions to cryptography, and sustained educational leadership.

▸ Claude Crépeau, for pioneering work on the foundation of oblivious transfer, two- and multiparty protocols, information-theoretic security, and quantum cryptography.

▸ Lars Knudsen, for contributions to the design and cryptanalysis of symmetric primitives and for service to the IACR.

▸ Hugo Krawczyk, for contributions to cryptography and technology transfer of cryptographic research results to secure Internet protocols.

▸ Victor S. Miller, for contributions to elliptic curve cryptography, pairing based cryptography, and the LZW compression algorithm.

▸ Rafail Ostrovsky, for contributions to the scientific foundations of cryptography.

DALE RECEIVES IEEE BOOTH AWARD

Nell B. Dale, one of the first women to earn a doctorate in computer science, was named 2013 recipient of the IEEE Computer Society Taylor L. Booth Award for her contributions to computer science education.

Dale, whose research research has focused on computer science education as an academic discipline, previously received the ACM SIGCSE Award for Outstanding Contributions to Computer Science Education, was the first woman to receive ACM's Karl V. Karlstrom Outstanding Educator Award, and is an ACM Fellow.

the enterprise side of the tech world," the report said.

To those who question the need for IT guest workers, Melland replies, "We do have a lot of qualified people, most of whom are employed, and companies can't find the exact skills they are looking for, so they feel they either need to bring those people here, or need to go outside the country to get them."

Melland says there is still a shortage of people with technical backgrounds and specific technology skills in certain metropolitan markets. "The unemployment rate overall is 7.5% and it is 3% for tech," he says, "but if you look at specific skills, it is quite low. The Bureau of Labor Statistics says for software developers, it is 1.8% across country; that includes some tech-intensive markets where it's practically zero. In more rural areas, it might be a little higher. But for network architects it is 2.2%; network and systems administrators, 1.1%." In these specific, highly demanded skill sets, there is more demand than supply, he says.

Yet Melland says the labor market for technology remains very tight, as evidenced by the low unemployment rate. Furthermore, average salaries have been rising substantially, he says. "The average tech salary across many skills is $85,000, way above the national average," he says, "which shows demand, and it has been going up several percentage points every year for the last three to four years. To get that, you have to offer a better package."

Salzman says IT salaries have stalled at 1998 levels, however, and that this is one indicator there is no shortage of qualified IT workers, because competition would have driven salaries up

Anecdotally, the Dice.com report noted that at the University of North Carolina-Charlotte, the number of IT undergraduates earning bachelor's degrees had spiked 41% year-over-year. The *Daily Pennsylvanian* reported last spring that the University of Pennsylvania was seeing overcrowding in its introductory computer science classes—the number of students had surged from about 50 in 2007 to 170 students this year—and included more than the usual number of students from the College of Arts and Sciences and the Wharton School.

Meland says there is still a shortage of people with technical backgrounds and specific technology skills in certain metropolitan markets.

The Harvard report articulates challenges and "hostile arguments" that humanists face nationally and internationally, and attempts to refute them. It presents the economic, cultural, social, scientific, vocational, and technological argument, as follows:

"Human societies, both literate and non-literate, have universally understood themselves through works of art that require deep immersion. In the twenty-first century, however, deep immersion is no longer the order of the technological day. New technologies disfavor the long march of narrative, just as they mitigate against sustained imaginative engagement. Students born after 1990 will not read paper books; much more significantly, they might not read books at all. The study of the 'deep-immersion' art forms is the study of shrinking, if not of dying, arts. Instead of lamenting that phenomenon, we should adapt to it. If we support the humanities, we should support media studies, not the study of the high arts."

The authors of the Harvard study offered their recommendations for bringing traditions into the 21st century "in a way that speaks to modern concerns." These include developing resources to attract freshmen to back to the humanities, creating more arts and exhibition spaces, obtaining funding for internships that allow undergraduates to experience a career in the humanities, investigating cross-school courses and co-teaching, and funding new faculty positions in the humanities.

James Simpson, a Harvard professor of English, said, "The humanities almost always tell us the same thing, that there is nothing new under the sun." Yet there are always new developments in computer science, which probably is part of what attracts an increasing number of students away from the humanities.

Further Reading

Brittany Ballenstedt
"America's Tech Talent Crunch 2013," Nextgov, May 14, 2013 (http://www.nextgov.com/cio-briefing/wired-workplace/2013/05/demand-it-grads-driving-supply-study-finds/63152/)

James Simpson and Sean Kelly
"In Brief: The Teaching of the Humanities at Harvard College," Harvard University Arts & Humanities Division, May 2013 (http://artsandhumanities.fas.harvard.edu/files/humanities/files/final_in_brief_mapping_the_future_may_22_from_mf.pdf)

James Simpson and Sean Kelly
"The Teaching of the Arts and Humanities at Harvard College: Mapping the Future," Harvard University Arts & Humanities Division, June 2013 (http://artsandhumanities.fas.harvard.edu/files/humanities/files/mapping_the_future_31_may_2013.pdf)

Hal Salzman, Daniel Kuehn, and B. Lindsay Lowell
"Guestworkers in the high-skill U.S. labor market: An analysis of supply, employment, and wage trends," Economic Policy Institute, April 24, 2013 (http://www.epi.org/publication/bp359-guestworkers-high-skill-labor-market-analysis/)

Hal Salzman
"What Shortages? The Real Evidence About the STEM Workforce," *Issues in Science and Technology*, Summer 2013 (http://www.issues.org/29.4/hal.html)

"Computing Degree and Enrollment Trends," CRA Taulbee Survey 2011-2012, Computing Research Association (http://cra.org/govaffairs/blog/wp-content/uploads/2013/03/CRA_Taulbee_CS_Degrees_and_Enrollment_2011-12.pdf)

"Humanities Indicators, a project of the American Academy of Arts and Sciences" (http://www.humanitiesindicators.org/content/hrcoII1.aspx)

Corydon Ireland
"Mapping the future: Reports tackle issues, concerns involving strengthening the humanities in a scientific age," *Harvard Gazette*, June 6, 2013 (http://news.harvard.edu/gazette/story/2013/06/mapping-the-future)

Karen A. Frenkel writes about science and technology and lives in New York City.

DOI:10.1145/2535913

Richard Heeks

Emerging Markets
Information Technology Impact Sourcing

New ways to contract IT work to base-of-the-pyramid suppliers.

INFORMATION TECHNOLOGY IMPACT sourcing is a new approach that clients can take when they outsource IT-related work to base-of-the-pyramid (BoP)[a] suppliers. In this column, I explain why impact sourcing is worthy of greater consideration and support from the IT professional and academic community.

Supply of IT work from the base of the pyramid is growing as diffusion of digital technology and attendant skills becomes almost universal. In theory, BoP supply would mean drawing IT workers from communities in developing countries averaging income levels of less than $2.50 per person per day. In practice, the idea covers a looser sense of IT staff that hail from low-income areas and/or have been excluded from mainstream employment opportunities.

Clients can select from a continuum of approaches to BoP outsourcing, as summarized in the accompanying figure:

▸ *Arbitrage outsourcing* seeks to bear down on wages and working conditions in order to minimize costs and maximize profits.

▸ *Mainstream outsourcing* is a conventional approach that reflects the steady diffusion of outsourcing suppliers from cities to large towns to small towns and beyond.

▸ *Ethical outsourcing* (also known as socially responsible outsourcing) takes mainstream outsourcing and requires that it meet certain minimum standards; typically relating to labor practices but also starting to include environmental issues.

▸ *Social outsourcing* (also known as developmental outsourcing) differs from ethical outsourcing as fair trade differs from ethical trade. Ethical outsourcing involves existing commercial players with either a commitment to or measurement of adherence to standards. Social outsourcing involves new non-market intermediaries who sit between the client and the BoP supplier.

The entire range of business process outsourcing activities can be encompassed by impact sourcing.

The key problem for the approaches shown on the left side of the continuum in accompanying figure—arbitrage especially but also some mainstream contracting—is sustainability.[1] They are associated with high levels of staff turnover and negative publicity about "digital sweatshops" and loss of IT jobs in the global North. From time to time, this blows up into public relations disasters for client firms and a high-profile reversal from offshoring to onshoring. In all cases, the bottom-line issue is the bottom line: these approaches can end up raising costs and failing to reap the benefits of low-wage IT staff in developing countries.

a The base of the pyramid, also known as the bottom of the pyramid, is an economics-derived term referring to the largest, but poorest, global socioeconomic group comprising approximately four billion people.

Tata Consultancy Services employees at the TCS campus in Chennai, India.

PHOTOGRAPH BY DHIRAJ SINGH/BLOOMBERG VIA GETTY IMAGES

So what is the impact sourcing alternative?

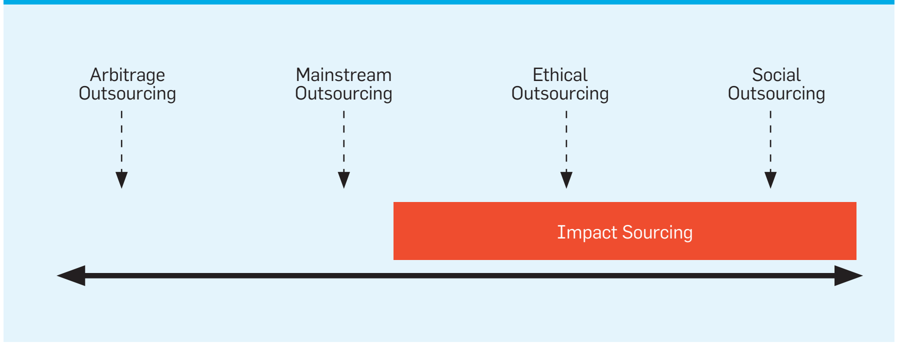
As shown in the figure, impact sourcing is a rather loose agglomeration of models, defined as "employing people at the base of the pyramid, with limited opportunity for sustainable employment, as principal workers in outsourcing…to provide high-quality, information-based services to domestic and international clients" and "to create sustainable jobs that can generate step-function income improvement." Its core elements then, are use of BoP workers and an intent that goes beyond simple contract fulfillment to the wider societal impact of the work being outsourced.

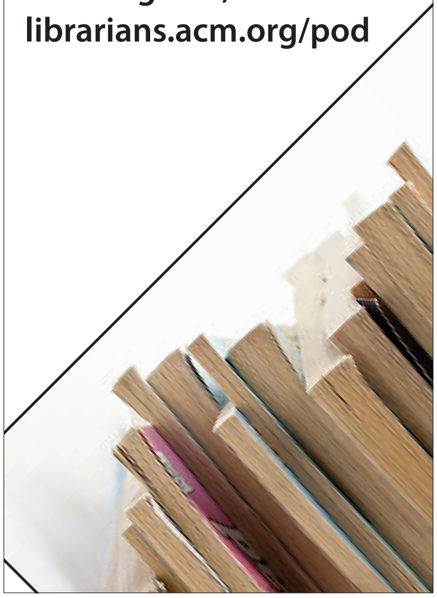
Impact sourcing received significant stimulus in 2011 following the release by the Rockefeller Foundation of its report *Job Creation Through Building the Field of Impact Sourcing*, which demonstrated this activity was already established in countries like India, South Africa, and Kenya.[6] (The preceding definitional quotes are taken from page 2 of that report.) Report authors estimated impact sourcing was already a $4.5 billion market employing 144,000 people and "has the potential to be a $20 billion market by 2015, directly employing 780,000 socioeconomically disadvantaged individuals."

The key impact sourcing players are clients, intermediaries (sometimes known as ISSPs: impact sourcing service providers) and suppliers. This is not charity, so the concerns of clients are those one would expect for any outsourcing: cost, delivery timescale, and quality of service. The heart of the value proposition tends to be cost rather than social impact. The latter gains emphasis for some clients, such as those in the public and non-profit sectors that have taken a lead in impact sourcing, but most clients are attracted by typical cost savings of 40% compared to conventional IT outsourcing routes.

Reflecting the continuum shown in the figure, intermediaries—the ISSPs who buffer the relationship between clients and BoP employees—vary. Some are fairly traditional players in the outsourcing market, looking for new, low-cost sources of IT labor and willing to adhere to corporate social responsibility requirements. Others—such as U.S.-based trailblazers Digital Divide Data and Samasource—are non-profits following the social outsourcing approach, for whom the so-

Continuum of approaches to outsourcing IT to the base of the pyramid.





cial mission is key. These high-profile intermediaries work across borders, bringing a developmental dimension to IT offshoring. But others work just within national borders; sourcing from urban clients to low-income urban and rural communities.

The sharp end of impact sourcing lies with the suppliers: the base-of-the-pyramid workers who deliver the IT services and who typically live in developing countries. One has to move beyond the slum dweller or peasant farmer stereotype to understand this group. Substantial numbers of the poor have high school diplomas or college, even university, degrees. Even those without these qualifications are increasingly making use of IT. So the entire range of business process outsourcing activities can be encompassed by impact sourcing. Contracts cover document and image digitization, image and video tagging, data entry and processing, call centers, translation, and more.

The wages paid to these employees are low by Western standards and below the average for their own national markets. But those points of comparison are largely irrelevant. Pay rates are significantly higher than averages in the local communities of these staff, even assuming there were other jobs they could take up, with estimates that impact sourcing increases their income by 80%–120%.[4] And, beyond this, impact sourcing has been shown to deliver benefits in terms of skills, attitudes, empowerment, social status, and connections.[3,5]

Of course, impact sourcing is not without its detractors; being in danger of a pincer movement of criticism from those against IT offshoring and those who disparage impact sourcing's wider analogues—ethical trade and fair trade. These critics tend to come from opposite sides of the political spectrum. Opponents of offshoring want government intervention to block free trade although they have been less vociferous about impact sourcing than conventional offshoring; perhaps because the former is still small and has the demonstrable value of bringing work to the world's poor.

Trade critics, by contrast, would argue that impact sourcing interferes with and undermines the due functioning of free markets, introducing inefficiencies and hampering development of a commercial IT sector within low-income locales. Ultimately, such debate is less about evidence and more about ideological position: Do you follow "the business of business is business" maxim or do you believe enterprise can and should be used to address some of society's ills?

As noted earlier, some clients may adhere to the first position, being unaware of or uninterested in the social dimension of impact sourcing and leaving the developmental intent of contracting to the ISSP. But more often clients have some level of social concern that ranges from a veneer of corporate social responsibility to a comprehensive social mission. Integrating that social mission into the outsourcing relationship is a challenge, and one that falls largely to the ISSP.[2]

At the pre-contract stage, developmental issues must be added in to due diligence. At the contract stage, social impact indicators may need to be developed and incorporated. Relationship governance often requires a longer-than-normal learning curve to bridge the gaps of expectations and worldview that exist between client, intermediary, and supplier. And performance management may demand the development of social return on investment techniques that are unfamiliar to the key players.

All of this makes impact sourcing somewhat different than conventional IT outsourcing models. There is a greater emphasis on learning and knowledge-building as all parties are, to some extent, venturing into the unknown. There is a greater concern with sustainability: a number of small ISSPs seem to have come and gone in the blink of a single contract. This has stranded their base-of-the-pyramid employees without a source of livelihood. And the relative novelty of the business model, the indirect nature of client-supplier relations, and the geographic distances often involved mean trust, image, and reputation matter greatly. ISSPs therefore have to focus heavily on their public relations profile.

The Rockefeller Foundation's support for impact sourcing and its initiatives such as Digital Jobs Africa are helping develop those profiles, but why should their intervention be necessary? Why aren't the bumper stickers of "doing good while doing well" or "give work not aid" enough to drive growth of this new IT model?

The answer is a series of challenges that must be addressed and which the market alone may not solve or may not solve in a foreseeable timeframe. From the perspective of demand, there is a long familiarity curve to be traversed. Vast numbers of potential clients are unaware of impact sourcing's existence (and the lack of immediate meaning in the terminology does not help). Some clients are half-aware but associate the base of the pyramid with backward villagers they would never consider as an outsourcing possibility. And some clients are three-quarters-aware but perceive too great a level of risk.

For the intermediaries, to the issue of sustainability noted earlier, we can add the challenge of scalability. Making a difference to the life of one poor person is a valid endeavor. But impact sourcing needs to make a difference to the lives of hundreds of thousands if it is to be taken seriously as a mechanism for development. ISSPs need help to grow beyond a cottage industry model.

And down at the BoP, there may be millions of possible employees but most are still on the wrong side of the digital divide in various senses. They might have a cellphone—a base that some ingenious ISSPs are using for impact sourcing—but the great majority will need broadband connectivity and substantial up-front training investments before they can find themselves on the right side of the divide, with the ability to participate in impact sourcing. Some models—such as those based around cellphones, or around breaking contracts into tiny "micro-work" packets, or those seeking to automate quality controls—will require further technical innovations before they can be mainstreamed to the poor.

Whether impact sourcing overall can be mainstreamed and fulfill its potential as a significant development tool is as yet unclear—it may remain a niche activity. But I cannot help noticing how very similar this all is to IT offshoring to India 30 years ago: A few brave companies had taken the plunge but the great majority of U.S. and European firms had not heard of the option, or laughed off India—which they saw as a land of bullock carts and maharajahs—as a possible IT location. Consider what happened next.

Down at the BoP, there may be millions of possible employees but most are still on the wrong side of the digital divide.

Impact sourcing is unlikely to be the next India as an IT sourcing model. But if it is to grow, it will need the support of the IT community. In part this means adding impact sourcing to the range of IT sourcing options considered. But it also means academic engagement. In the social sphere, all those involved need guidance on business models, best practices, and impact evaluation. In the technical sphere, new technologies are still needed to improve sustainable, high-quality access to digital tools for those at the BoP; and to deliver fast, contextually relevant IT training. If this support materializes then so, too, may impact sourcing's developmental promise.

References

1. Babin, R. and Nicholson, B. Corporate social and environmental responsibility and global IT outsourcing. *MISQ Executive 8*, 4 (Dec. 2009), 123–132.
2. Heeks, R. A model for assessing IT impact sourcing relationships. *ICTs for Development* (Sept. 27, 2012); http://ict4dblog.wordpress.com/2012/09/27/a-model-for-assessing-it-impact-sourcing-relationships/
3. Heeks, R. and Arun, S. Social outsourcing as a development tool. *Journal of International Development 22*, 4 (May 2010), 441–454.
4. Kubzansky, M., Cooper, A. and Barbary, V. *Beyond Aid.* Monitor Group, Boston, 2011.
5. Madon, S. and Sharanappa, S. Social IT outsourcing and development. *Information Systems Journal 23*, 5 (Sept. 2013), 381–399.
6. Rockefeller Foundation. *Job Creation Through Building the Field of Impact Sourcing.* Rockefeller Foundation, New York, 2011.

Richard Heeks (richard.heeks@manchester.ac.uk) is Director of the Centre for Development Informatics at the University of Manchester, U.K.; http://www.cdi.manchester.ac.uk/.

Calendar of Events

January 9–11
The 8th International Conference on Ubiquitous Information Management Communication
Siem Repa Cambodia,
Sponsored: SIGAPP,
Contact: Sukhan Lee,
Email: ish@ece.skku.ac.kr

January 12–14
Innovation in Theoretical Computer Science
Princeton, NJ,
Sponsored: SIGACT,
Contact: Bernard Chazelle,
Email: chazelle@cs.princeton.edu
Phone: 609-258-5380

January 20–23
19th Asia and South Pacific Design Automation Conference
Singapore,
Sponsored: SIGDA,
Contact: Yajun Ha,
Email: elehy@nus.edu.sg

January 22–24
The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages
San Diego, CA,
Sponsored: SIGPLAN,
Contact: Suresh Jagannathan,
Email: suresh@cs.purdue.edu

February 15–19
Computer Supported Cooperative Work
Baltimore, MD,
Sponsored: SIGCHI,
Contact: Susan R. Fussell,
Email: sfussell@cornell.edu
Phone: 607-255-1581

February 22–26
ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming
Orlando, FL,
Sponsored: SIGPLAN,
Contact: Jose E. Moreira,
Email: jmoreira@us.ibm.com
Phone: 914-525-6267

February 23–25
The 2014 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays
Monterey, CA,
Sponsored: SIGDA,
Contact: Vaugn Timothy Betz,
Email: vaughnbetz@gmail.com
Phone: 416-766-2197

DOI:10.1145/2535914

David Anderson

Historical Reflections
Patrick Blackett: Providing 'White Heat' to the British Computing Revolution

Reflections on a Nobel Prize-winning physicist's early contributions to computing.

HERMAN GOLDSTEIN, ONE of the early developers of ENIAC, once remarked about the ENIAC project "…there was a lot of controversy about who did what but I think the fact is that…there's really about an infinite amount of credit to be distributed and therefore everybody who contributed I think did a remarkable job."[4] Within the history of British computing there is a long-established tradition, not well supported by the available historical evidence, of attributing almost all of the credit for the development of the Manchester Baby, to the engineers F.C. Williams and (to a greater extent) Tom Kilburn, while substantially ignoring contributions from other sources.[a] Anyone who has worked on even relatively modest projects, within an institutional context, will understand perfectly well, that there is a great deal more involved in seeing matters to a successful conclusion, or getting them off the ground in the first place, than simply "doing the science."

Concentrating exclusively on a few individuals, can lead us away from a deeper understanding of the political, economic, and wider scientific context in which landmark developments took place. For example, Patrick Maynard Stuart Blackett—Baron Blackett of Chelsea—is best remembered as an outstanding and versatile physicist. He was the Nobel laureate for physics in 1948 and served as a key scientific advisor to the governments of Britain and India. He enjoyed an academic career at Cambridge University, Birkbeck College, the University of Manchester, and Imperial College London, and is widely recognized as having had a significant impact on fields ranging from particle physics to continental drift.

Far less well known is the hugely influential role Blackett played in the early history of British computing. Far from the center of his own research in-

Patrick Blackett standing by equipment in the Cavendish Laboratory circa 1930.

PHOTOGRAPH FROM AIP EMILIO SEGRE VISUAL ARCHIVES, COURTESY LAWRENCE BADASH

a For example, Lavington, S.H. *A History of Manchester Computers* (2nd ed.). British Computer Society, 1998.

terests, he actively shaped the careers and facilitated the projects of many of the pioneers who are most closely associated with the development of the computer. Through his intervention, significant resources were made available without which the computing landscape in post-war Britain would have looked very different indeed.

Blackett was born on November 18, 1897, the son of Arthur Stuart Blackett and Caroline Frances Maynard. He attended a small preparatory school before taking up a place at the Osborne Naval College where the curriculum accentuated engineering, as well as theoretical and practical science. Students were given encouragement to develop skills such as the use of tools, the operation of lathes, turning, and forging. This proved to be an ideal grounding for Blackett, whose future scientific career would be characterized by mastery of experimental practice, built on a profound theoretical understanding. In 1914, after two years at Osborne, and a similar period at Dartmouth College, Blackett joined the Royal Navy as a midshipman on the HMS *Carnarvon*. By the end of hostilities, Blackett had risen to the rank of lieutenant, had served on a variety of different vessels, and had seen action in a number of battles including the Falkland Islands and Jutland.

In 1918, Blackett was one of a group of 400 junior officers chosen by the Royal Navy to attend a six-month short course at Cambridge. However, soon after arriving at Magdalene College, Blackett visited the Cavendish Laboratory and was so impressed by what he saw that he resigned from the Navy in order to become an undergraduate.

Over the following two decades, Blackett's career made rapid progress. In 1933, Blackett was elected a Fellow of the Royal Society and, that same year, left Cambridge and moved to Birkbeck College, London, to head his own laboratory. In autumn 1937, Blackett succeeded W.L. Bragg as the Langworthy Professor of Physics in the Victoria University Manchester and right away set about refocusing the department's profile and activities. As part of the overall reorganization, he persuaded the university to create a new chair in theoretical physics, installing Douglas Hartree, who had previously held the chair in applied mathematics, as the first incumbent.

After the war, Blackett was eager to see a civilian computer developed at Manchester.

It was at this point that Blackett's personal involvement with computing seems to have begun. The stimulus came from his interest in the mechanical differential analyzer that Douglas Hartree and Arthur Porter had installed in the basement of the physics department. This was an analog device in which the value of a variable was determined by the rotation of a shaft, which was, in turn, dependent on a human operator accurately tracing an input curve. Working together with F.C. Williams, who was, at the time, an assistant lecturer in the Department of Electro-Technics, Blackett devised a photoelectric curve follower,[2] capable of much greater accuracy than had been possible working by hand.

Williams notes that it was Blackett who first introduced him to mechanical computation: "…my first connection with computers of any kind was just before the war, when…Professor Blackett asked me to make an automatic follower for feeding data into this computer. This was a very interesting and fruitful piece of work as it gave me an introduction to servo-mechanisms and brought me into the concept of computation for the first time using mechanical aids of some kind."[5]

Williams' engineering expertise impressed Blackett sufficiently for Blackett to "channel" him into war work as part of the Royal Air Force radar research group at Bawdsey Research Station—one of the precursors to the Telecommunications Research Establishment.

Another significant figure on the postwar British computing scene, whose choice of war work Blackett influenced, was Maxwell H.A. Newman.[1] Writing to the director of naval intelligence, Rear Admiral John Henry Godfrey, on May 13, 1942, Blackett recommended Newman for a job at Bletchley Park describing him as "one of the most intelligent people I know."[b]

After the war, Blackett—one of the coterie of people who were aware of the existence of the Colossus—was eager to see a civilian computer developed at Manchester.[c]

Blackett's support was crucial because his standing in the scientific community assured the project would face little effective opposition. Blackett's role initializing, orchestrating, and enabling the work was mostly carried out in the background, but was of such importance that it is reasonable to think of him as the prime mover and first architect of the Manchester computing phenomenon. Without his involvement, it is by no means an exaggeration to say that Manchester would have played no significant role in the development of first-generation digital computers.

As it happened, there was a vacancy for the Fielden Chair of Mathematics at the University of Manchester, and Blackett was resolved that Newman should apply for the position and, once appointed, should lead the computing building project. This was not a plan which found much favor with Newman's wife Lyn, who was absolutely appalled at the prospect of leaving the family home in Cambridge "for the perpetual gloom of Manchester."[d] Her resistance was overthrown only when "Patrick got at that always sensitive place, pride in a husband's career—he said if Max chose to take a back seat in Cambridge still, another would gladly step in."[e]

b Personal correspondence from P.M.S. Blackett to M.H.A. Newman, June 22, 1942 Facsimile available at The Newman Digital Archive, the Future Proof Computing Group, Portsmouth, and St. John's College, Cambridge; http://bit.ly/17No0r0

c Brian Randell, personal communication with author, January 20, 2005.

d Letter from Lyn Newman to Hella Weyl, March 1945. Cited in W. Newman, "Married to a Mathematician: Lyn Newman's Life in Letters," *The Eagle*, St. John's College, Cambridge, 2002, 47–55.

e Letter from Lyn Newman to Antoinette, Viscountess Esher, 1948. Cited in Newman, W., "Married to a Mathematician: Lyn Newman's Life in Letters," *The Eagle*, St. John's College, Cambridge, 2002, 47–55.

Having thus secured Newman's services, Blackett would certainly have been active in ensuring that the university supported the development of a Manchester computer, but the vice chancellor insisted "that the financial support must come from outside"[f] so, at Blackett's instigation and with his encouragement, Newman applied to the Royal Society for funding.

Initially, Newman's bid met with opposition from Charles Darwin who, mindful of the National Physical Laboratory's plans to develop its own computer, saw no reason to establish a second (rival) project at Manchester. A committee comprising Darwin, Hartree, William Hodge, Henry Whitehead, and Blackett was established to settle the matter, and approved Newman's proposal on a majority vote with a single dissenting voice.

Neither Blackett nor Newman had either the inclination, or the skills, to undertake the detailed hardware design of a computer. Therefore, it was essential to procure an engineer who could lead the hardware development. There was some limited support available from the head of Electro-Technics, Willis Jackson, who was prepared to provide a departmental home for the engineer(s) once appointed, but had no intention of personally leading the circuit design.

Blackett put it to Newman that Williams might be the man for the job, but in the post-war period, capable engineers were in very short supply, and Williams was also being pursued by NPL, which hoped Williams would build Alan Turing's ACE (Automatic Computing Engine). In the end, the prospect of a professorial appointment, headship of a university department, and the freedom to pursue his own ideas for computer memory were sufficient inducement to bring Williams to Manchester.

Just a few months after the first successful running of the Manchester Baby machine, Blackett suggested that the Ministry of Supply's chief scientist, Ben Lockspeiser, who was paying an informal visit to Blackett, should see the computer for himself. This took place in October 1948. A junior engineer, Geoff C. Tootill, was on hand to give what turned out to be a very successful demonstration of the machine. Lockspeiser was sufficiently impressed with what he saw to arrange immediately for the treasury to commit £100,000 to support Ferranti, a local manufacturer, in the further development of a computer built under Williams' overall direction. The creation of this link between the university and industry was an important factor in assuring the future of computers at Manchester.

f M.H.A. Newman, letter to the Secretary of the Royal Society, January 28, 1946. Facsimile available at The Newman Digital Archive, the Future Proof Computing Group, Portsmouth, and St. John's College, Cambridge; http://bit.ly/14LmrMe

During his long and varied career, Blackett was the deserved recipient of almost every honor his profession and colleagues could bestow.

Blackett's roles on a variety of government advisory committees placed him in an excellent position to have an impact on the way in which computing developed and was supported in Britain, and was instrumental in setting up the National Computing Centre, the purpose of which was to be the voice of the computer user, to give advice and training to computer users outside the public sector, to act as a library service for existing software, and to develop new software.

During the 1940s and 1950s, Blackett's outspoken views about British and American nuclear weapons policies led the author George Orwell to include him on a blacklist of 38 crypto-communists or fellow travelers that Orwell drew up for the British Foreign Office. However, when Harold Wilson became leader of the Labour Party in 1963, Blackett's political fortunes recovered. It was at Blackett's suggestion that Wilson created the Ministry of Technology (MinTech) which became the most comprehensive production ministry Britain has ever had,[3] and initially, according to Maddock, Blackett's views "were accepted as absolute and his priorities determined the activities of the day."[g]

Between 1965 and 1970, Blackett served as president of the Royal Society, and in 1969 he was created a life peer taking the title Baron Blackett of Chelsea. In 1947, Jawaharlal Nehru sought his opinions on the research and development needs of the Indian armed forces. Over the course of the following 20 years, Blackett's interest in the subcontinent deepened, and he gradually reined back on his involvement with MinTech. He visited India on many occasions, often as Nehru's guest.

Blackett died on July 13, 1974, at 76 years old. During his long and varied career, he was the deserved recipient of almost every honor his profession and colleagues could bestow. Sir Harrie Massey voiced the opinion of many when, speaking at a memorial gathering in Blackett's honor, he observed that Blackett: "Never spoke without having something stimulating to say. The fact that he was by no means taciturn is a tribute to the fertility of his mind and the width of his interests."[6] C

g Personal correspondence with Sir Ieuan Maddock (deputy controller in the Ministry of Technology, 1965) cited in Lovell, "Patrick Maynard Stuart Blackett, Baron Blackett of Chelsea," 1–116.

References

1. Anderson, D. Max Newman: Forgotten man of early British computing. *Commun. ACM 56*, 5 (May 2013), 29–31.
2. Blackett, P.M.S. and Williams, F.C. An automatic curve follower for use with the differential analyser. In *Proceedings of the Cambridge Philosophical Society 35*, 3 (July 1939), 494–505.
3. Edgerton, D. The "white heat" revisited: The British government and technology in the 1960s. Twentieth Century British History 7, 1 (1996), 53–82.
4. Evans, C.R. Interview with Herman Heine Goldstine, Unpublished interview (transcript by David P. Anderson). Science Museum/National Physical Laboratory, 1976.
5. Evans, C.R. Pioneers of computing 7: F.C. Williams. Interview with F.C. Williams audio recording 1976 (transcript by David P. Anderson). Science Museum, London, 1998.
6. Hodgkin, A. et al. Memorial meeting for Lord Blackett, OM, CH, FRS, at the Royal Society on October 31, 1974. *Notes and Records of the Royal Soc. of London 29*, 2 (1975), 135–162.

David Anderson (cdpa@btinternet.com) is the CiTECH Research Centre Director at the School of Creative Technologies, University of Portsmouth, U.K.

DOI:10.1145/2535915

Peter J. Denning

The Profession of IT
Design Thinking

Design thinking is the newest fashion for finding better solutions to problems. Combining it with computational thinking offers some real possibilities for improving software design.

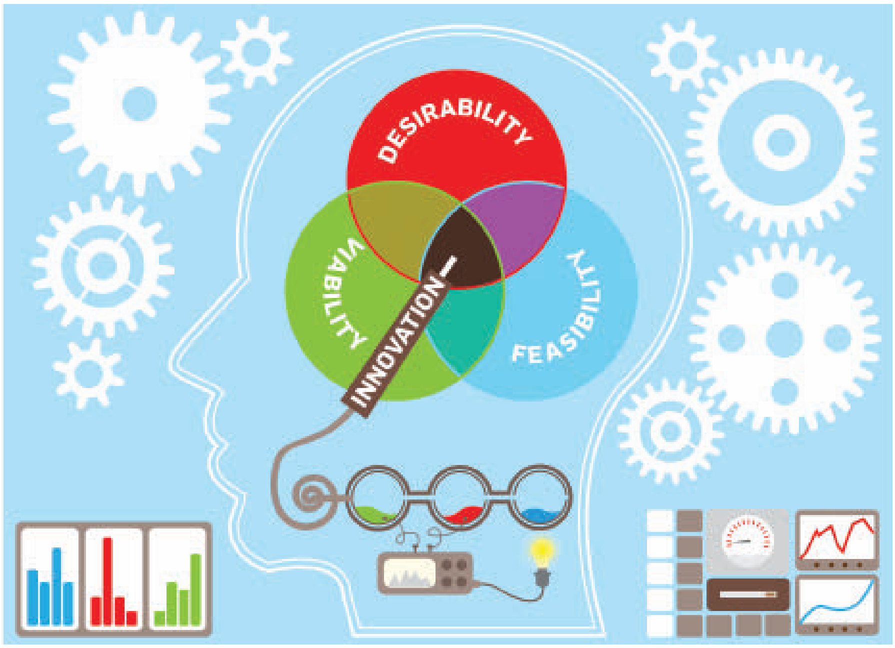


IMAGE COLLAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

HAVE YOU NOTICED design thinking? In recent months a spate of news reports, *60 Minutes* segments, and TV news reports have told how the IDEO Company and its founder, David Kelley, have developed an innovative new approach to product design. Many social service and government organizations are now looking at IDEO's design thinking as a path to process innovation in their organizations. Practitioners trying to solve wicked problems are also finding fruit in design thinking. I would like to share some reflections on these reports.

I will review computing's long history of involvement in design. Some of our software development communities, especially those under the headings of agile, participatory, and user-centered, have been design thinkers for a long time. I will offer a note of caution about the claim of some media advocates that design thinking will speed up the processes of innovation.

Design

Design is familiar in many fields including fashion, products, architecture, engineering, science, and software development. Design is a process where we create and shape artifacts that solve problems. In software, for example, design means crafting software that does jobs users want done. Software designers intentionally support practices, worlds, and identities of the software's users. Designers have accumulated much practical wisdom that is expressed with design principles such as separation of concerns, modularity, abstraction, layering, wholeness, utility, resiliency, beauty, and timelessness. Design principles in computing guide us to ways of building machines whose behaviors are useful and meaningful in their user communities.

Seasoned designers constantly run experiments with prototypes to learn how well the machines might work and how users might react to them. Maurice Wilkes stressed this point in his 1967 ACM Turing Lecture,[5] saying that a great strength in the early days was the willingness of research groups to construct experimental computers without necessarily intending them to be prototypes for commercial production. Their experiments produced a body of knowledge about what would work and what would not work. In his 1995 memoir, Wilkes strongly criticized the more recent trend to ignore the historical development and try to design from scratch, as in the personal computer world. Without the knowledge of what worked and what did not, designers have tended to repeat the same mistakes.[6] Like Fred Brooks (author of *No Silver Bullet*), Wilkes believed that good design is a skill set with many dimensions, well worth cultivating.

Since its beginnings in the 1940s, software has had a reputation of being extremely error prone. Programmers have always been frustrated by the amount of time they need to spend locating mistakes in their own programs, and in protecting software and data from external errors. This has not been easy given the vulnerabilities inherent in the chain of transformations that designers must master (see the accompanying figure):

1. The specification does not accurately represent the designer's intention, or is based on misunderstandings of user expectations.

2. The programmer makes mistakes, for example, by introducing bugs or approximations that cause the software to violate its specifications. Moreover, the compiler might contain bugs or Trojan horses that cause the machine code to be not equivalent to the source program.

3. The machine itself contains bugs, defects, malfunctions, intrusions by other buggy machines or attackers, and other factors that cause it to misbehave while executing its basic operations.

4. The users' expectations of what the machine's job is differs from what they see the machine doing.

When combined with the general concern for getting work done on time, these error sources have led to five traditional criteria for software design:

▸ *Requirements—Does it have a clear purpose?* The designer knows what job the machine is intended to perform and can state the requirements precisely as a specification. Articulating requirements is a challenge because interviewing the intended users about what they want is notoriously unreliable.

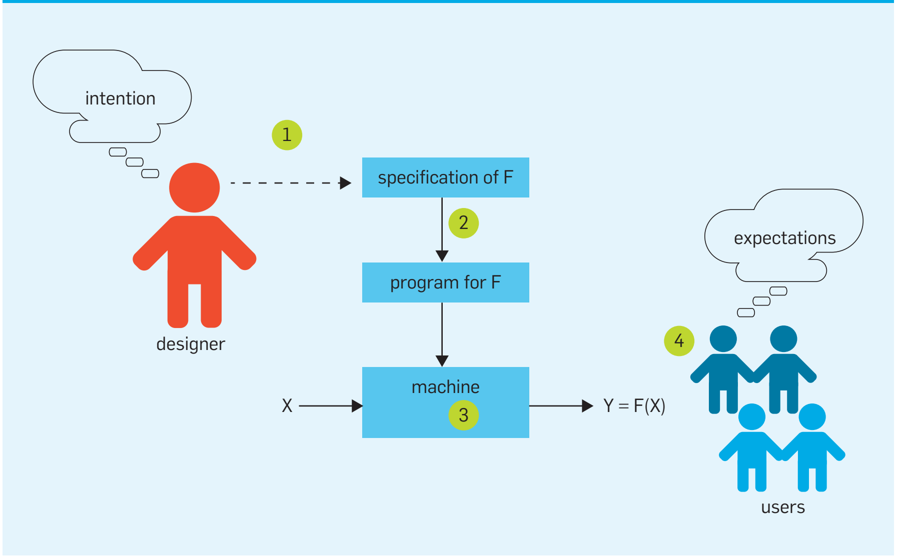
▸ *Correctness—Does it work properly?* The behavior of a source code program provably meets precise specifications. Correctness is challenging because requirements are often fuzzy and proofs are often computationally infeasible. Experimental methods are often the only practical way to learn user requirements, arrive at precise specifications, avoid intractability, and test prototype behavior.

▸ *Fault tolerance—Does it keep working?* The software and its host systems can continue to function despite small errors, and will refuse to function in case of a large error. Redundancy supports fault tolerance by duplicating hardware and data so that a failed component can be bypassed by other still-working components and datasets. Error confinement supports fault tolerance by structuring the operating environment so that no process has access to any object other than those it needs for its computation and by limiting (or eliminating) the super user state.

▸ *Timeliness—Does it complete its work in time to be useful?* The system completes its tasks within the expected deadlines. Supporting techniques include algorithm analysis, queueing network analysis, and real-time system deadline analysis.

▸ *Fitness—Does it align well with the user environment?* Fitness is challenging because assessments like dependability, reliability, usability, safety, and security are context sensitive and much of the context is not obvious even to the experienced designer. The famous architect Christopher Alexander advocated an approach to design that was immersed in the context of how dwellers used buildings.[1] Alexander's work inspired a group of software designers to found a "software pattern community" devoted the ideals of good software design. A software pattern characterizes a large number of situations that a programmer is likely to encounter, and offers guidance on how to structure the program to best fit the pattern. The pattern community appeals to my sense of empiricism because they are relentless about testing ideas with potential users and learning from the feedback. A good introductory book on the topic is *Pattern Languages of Software Design* (Addison-Wesley, 1995), by James Coplien and Douglas Schmidt. Coplien is one of the founders of the pattern community.

Four sources of errors in computations.



Software and Computer System Design

Despite their best efforts with tools, languages, and project management, software experts felt they could not keep up with the demands for ever-larger reliable systems. In 1968, they founded the field of software engineering to apply standard engineering methods to meet two concerns:

1. Increasing reliability and managing risk by eliminating or confining errors.

2. Seeing software projects through to completion (delivery) and subsequent evolution (maintenance).

Within computing various schools of thought have developed around specific approaches to these concerns. These schools have advanced "process models" like waterfall or spiral, or "design approaches" such as participatory, user centered, agile, or pattern. They are all after the same thing, but they weigh the criteria in different ways. Barry Boehm argued that the standard engineering design approach of careful, almost rigid process was at the strict end of a planning spectrum and agile methods were at the flexible end.[2] He thought that careful planning is needed when reliability, safety, and security are important and that agility is needed when usability and evolvability are important. He exhorted the careful planning and the agile schools to collaborate on finding a common ground for better systems.

Design Thinking

Design thinking means to intentionally focus the design around the concerns, interests, and values of the users. The current strand of design thinking that has captured much public attention and interest originated with industrial product design in the company IDEO founded in 1991 by David Kelley in partnership with several other design firms.

In 2006, Kelley founded the Stanford Design Center, which has become an intellectual center for a design thinking movement. The IDEO philosophy emphasizes that design is a team sport with three principal values:

1. Many eyes—Design teams include diversified expertise such as engineering, human factors, communication, graphics, ethnography, sociology, and more. Each team member's unique perspective helps the other members see things they would not ordinarily see.
2. Customer viewpoint—Design teams visit customer places to interview them and watch what they actually do, including their reactions in extreme "stress" cases.
3. Tangibility—Design teams build prototypes and mockups, try them out, and learn from the feedback and reactions.

There is a big interest in the public sector to apply design thinking to its own problems; the standard government approach of one-size-fits-all does not work for the diverse communities agencies are trying to serve. IDEO has helped agencies improve their processes, often with excellent results that receive a lot of publicity.

Design thinking has also been helpful in addressing wicked problems.[4] Design teams generate moods of collaboration, often leading to breakthroughs for resolving their wicked problems. When the design team brings together all the various stakeholders of a company, they are often able to win the commitments from multiple divisions of the company to see new ideas through to production. It should be noted that design thinking is not the only successful method for generating collaboration: so also do Appreciative Inquiry, Charrettes, and the Straus Method.

Design thinking is already deeply embedded into the software pattern community.

Design and Innovation

I am wary of the claim that design thinking speeds up innovation. Design and innovation are related, but not the same. Innovation is concerned with getting a community of people to adopt a new practice often organized around an artifact or process created by a designer. The only time design thinking might accelerate innovation is when the ideation stage has been blocked by lack of ideas or by conflicts among key stakeholders.

Master designer Don Norman puts this point differently.[3] He says designers working alone have a low rate of technology adoption. Technological entrepreneurs pursue the business opportunities and work hard, often over many years, to win the commitments for adoption. Design and entrepreneurship are both needed to transform new proposals into adopted practices.

Disruptive innovations, which reconfigure entire communities, are quite rare. When they do happen, they seem spectacular. Well-designed artifacts are often cited as the causes of the innovation. But this impression seldom holds up under scrutiny. Consider the iPhone. The iPhone is partly a story of design (Steve Jobs had help from IDEO with the artifact) but it is mostly a story of entrepreneurship: Apple miniaturized components, created portable apps, put many sensors into the phone, generated a community of software app developers, cloned the Apps Store from the successful iTunes store, partnered with AT&T and later Verizon, created a new class of data plans for phones, and built a new operating system—iOS—to support it all. Apple caused the innovation. The iPhone was the tip of an iceberg of practices and business arrangements that made it work.

Seasoned innovators work with a "90% rule"—90% of the work in achieving an innovation goes into the adoption phase. Ideation, where design thinking produces its value, is the other 10%. But many media reports would have you believe that design thinking gets you 90% of the way to innovation. I take issue with these pundits. While a collaborative team can get things done faster, often the design team is not a collaborator with the production, marketing, PR, and community outreach divisions of a company.

Putting It All Together

Design thinking calls attention to creativity and imagination in the ideation process, emphasizing collaborative, diverse, customer sensitive design teams. It also emphasizes frequent customer feedback from prototypes that elicit their reactions. Design thinking is already deeply embedded into the software pattern community, which is part of agile software development. That community has accumulated a large set of insights into what makes for successful software design. You need look no farther than that community to see how to put design thinking to work in software development.

Computing's traditional view of design is strongly flavored by its concern for building artifacts that are error tolerant or error free. Design thinking is strongly flavored by its concern for understanding what job an artifact does for its users. If the two kinds of thinking were blended together, some significant advances in software design and development would surely follow. **C**

References

1. Alexander, C. *The Timeless Way of Building.* Oxford University Press, 1979.
2. Boehm, B. Get ready for agile methods, with care. *IEEE Computer* (Jan. 2002), 64–69.
3. Norman, D. Technology first, needs last: The research-product gulf. *ACM interactions 17*, 2 (Mar.+Apr. 2010).
4. Roberts, N. Wicked problems and network approaches to resolution. *The Int. Public Mgmt. Review 1*, 1 (2000).
5. Wilkes, M. Computers then and now. (1967 Turing Award lecture). *JACM 15*, 1 (Jan. 1968), 1–7.
6. Wilkes, M. *Computing Perspectives.* Morgan Kaufman, San Francisco, CA, 1995.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of ACM *Ubiquity*, and is a past president of ACM. The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

DOI:10.1145/2535916

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
A Lesson in Resource Management

Waste not memory, want not memory—unless it doesn't matter.

Dear KV,
I have been reworking a device driver for a high-end, high-performance networking card and I have a resource allocation problem. The devices I am working with have several network ports, but these are not always in use; in fact, many of our customers use only one of the four available ports. It would greatly simplify the logic in my driver if I could allocate the resources for all the ports—no matter how many there are—when the device driver is first loaded into the system, instead of dealing with allocation whenever an administrator brings up an interface.

I should point out that this device has a good deal of complexity and the resource allocation is not as simple as a quick malloc of memory and pointer jiggling—a lot of moving parts are inside this thing.

We are not talking about a huge amount of memory by modern standards—perhaps a megabyte per port—but it still bothers me to waste memory, or really any resource, if it is not going to get used. I am old enough to remember eight-bit computers with 64 kilobytes of RAM, and programming those gave me a strong internal incentive never to waste a byte, let alone a megabyte. When is it OK to allocate memory that might never be used, even if this might reduce the complexity of my code?

Fearful of Footprints

Dear Footprints,
The answer to your question is easy. It is sometimes OK to allocate memory that might never be used, and it is sometimes not OK to allocate the same memory. Ah, are those the screams of a programmer without a black-and-white, true-or-false answer to the question that I hear? Delightful!

Software engineering, much to your and my chagrin, is the study of tradeoffs. Time vs. complexity, expediency vs. quality—these are the choices we deal with every day. It is important for engineers to revisit their assumptions periodically, perhaps every year or two, as the systems we work on change under us quite quickly.

Programmers who are paying attention to the systems they use—and I know that each and every one of my readers is paying attention—have seen these systems change dramatically over the past five years, just as they had the five years before that, and so on, back to the first computers. While processor frequency scaling may have paused for the moment (and we will see how long that moment lasts), the size of memory has continued to grow. It is not uncommon to see single servers with 64 and 128 megabytes of RAM, and this explosion of available memory has led to some very poor programming practices.

Blindly wasting resources, such as memory, really is foolish, but in this case it is not an engineering tradeoff, it is an example of a programmer who is too far from their machine trying to "just make it work." That is not programming, that is just typing. Software engineers and programmers worth their expensive chairs and high salaries know they do not want to waste resources, so they try to figure out what the best- and worst-case scenarios are and how they will affect the other possible users of the system. Users in most cases are now just other programs, rather than other people, but we all know what happens to a system when it starts to swap things out of memory onto secondary storage. That's right, your DevOps people call you screaming at 3 A.M. Screaming people are never that much fun, except at a concert.

You mentioned this software is for a "high-performance" device, and if by that you mean it goes in a typical 64-bit server-class machine, then no one is really going to notice a megabyte, or four, or even eight. A high-end server-class machine is unlikely to have less than four gigabytes of RAM. Even if you allocate four megabytes at system startup time, that is one-tenth of 1% of the available RAM. People writing in Java will suck down far more than that just starting their threads. Are you really going to worry about less than one-tenth of a percent of memory?

If you had told me that this driver was for some limited-memory-size embedded device, I would give other advice, since that system might not have 4GB of RAM; but then again, given what most phones and tablets have in them now, it might.

People are often right when they say, "Waste not, want not," but it is also important to take your moderation in moderation.

KV

Dear KV,

I am converting a Web application to run on a mobile platform. While the application does not handle banking information or anything with crazy security like that, it does still require the user to type a password. Our password requirements are not too strict, though we do have a minimum size of eight characters and require one uppercase letter and one nonalphanumeric character. Because on-screen typing is so inaccurate, our product development folks want us to relax our password requirements even more, allowing the user to have a four-character, all-lowercase password, which they call a mobile PIN code. The mobile PIN would work only from the mobile app and not on the Web. I have tried to explain to the product development group that four characters simply are not enough, but maybe if we are restricting this to the mobile app, it will be OK. What do you think?

Pinned Down

Dear Pinned,

I have been wondering when someone would write a letter like this. Having typed on a variety of tablets in the past few years, I knew it was only a matter of time before some marketing or product-type person would ask an engineer to dumb down security for convenience. I am glad you pointed out your application does not relate to banking, as I would have had to write back immediately and ask, "Which bank?!", and that could only lead to trouble.

As a quick aside, I do find it interesting that the world thinks only of banks when they think of security. While it would be very bad for someone to transfer all the money out of your bank account into their own, the fact of the matter is that a lot of really bad things can happen online that do not relate directly to cash flow. Weak passwords can leave users open to identity theft, stalkers, kidnappers, and their exes. It is up to you to think about which of those is better or worse than losing some cash. I would take losing money over being stalked by my exes.

The problem with a four-character password is that it is too short and easy to guess.

The problem with a four-character password—as you point out—is that it is too short and easy to guess. A PIN with a card, such as a card used at an ATM, is employed because it requires physical possession of something, the card, to be effective. Schemes for protecting users online, such as those involving passwords, depend on the user presenting a combination of something they have, something they are, or something they know. A password is an example of the latter. The schemes can be mixed and matched, such as the PIN and a physical card, where the user has something and knows something.

The problem with current mobile devices is that they are much more limited in their ability to handle user input than, say, a computer with a keyboard. On-screen keyboards are not very good, which is probably why someone at Google thought to use a pattern for device unlocking on the Android. I find that system a bit silly and easy to shoulder surf, but it is good to see someone trying to do something differently.

It would be nice to pretend that the device itself would be proof of something the user has, but since the point of the password is to prevent a malicious party from accessing the user's data after the device has been lost or stolen, it needs to be sufficiently strong to deter an attacker. If you are unable to fight back on password complexity, it is time to break out the lockout option. A four-digit pin code is mostly a problem if you allow the attacker a large number of attempts to guess the PIN. If, after three tries, you lock out the user for five minutes, and then let the user try again, it is going to take a long time for the attacker to try enough PINs to guess the right one—that is, if the user has not picked a common PIN, such as 1234 or 2580 (an exercise for readers is figuring out why that second code is so common; for a more academic study of the problem of PINs see the paper, "A birthday present every eleven wallets? The security of customer-chosen banking PINs," by Joseph Bonneau et al from the Computer Laboratory of the University of Cambridge at http://www.jbonneau.com/doc/BPA12-FC-banking_pin_security.pdf).

Being the kind of person I am, I also like the idea of three failed tries causing the device not to work at all, including erasing all local data and then requiring the user to go through a recovery flow that does not involve the device. If your application is used by people who are often mentally impaired in some way (and no, I do not mean they are upper management, but I do mean when it is used for social networking), then an annoying recovery system is not going to get past your product development folks. Too many people want to upload pictures of their friends in compromising positions, and that, alas, requires compromising on security.

KV

Related articles on queue.acm.org

Power-Efficient Software
Eric Saxe
http://queue.acm.org/detail.cfm?id=1698225

Network Virtualization: Breaking the Performance Barrier
Scot Rixner
http://queue.acm.org/detail.cfm?id=1348592

Kode Vicious: The Doctor Is In
George Neville-Neil
http://queue.acm.org/detail.cfm?id=1105672

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

DOI:10.1145/2535917

Wendy M. DuBow, Ruthe Farmer, Zhen Wu, and Malia Fredrickson

Broadening Participation

Bringing Young Women into Computing Through the NCWIT Aspirations in Computing Program

A program to encourage and support girls and women in pursuing their computer science interests.

WOMEN HAVE ACHIEVED parity, or majority, in many previously male-dominated fields, including law, medicine, business, and biology, but not (yet) in computing and engineering. We know that women are just as capable as men at succeeding in these fields. At this point, young women often need individual encouragement to pursue computing. The National Center for Women & IT (NCWIT) is supporting women's entry and persistence in this field, while at the same time helping to create academic and work environments that are egalitarian and welcoming to them. One avenue NCWIT is pursuing is the Aspirations Computing program, which identifies and supports girls in their computing interests and includes an award honoring high school girls for their computing-related achievements. The program also includes a growing number of other components and activities, as described in this column.

Research shows that encouragement helps individuals persist in the face of adversity.[2] Individual encouragement is essential to retention when girls and women express doubts about whether they belong in computing. Women report more often than men that they entered computer science because of a teacher, family member, or friend's encouragement.[1] Support can make a big difference in a girl's belief that she is competent enough to succeed at computing tasks, which can lead to persistence in the field.

The Problem

Being female in a male-dominated field is not easy. Girls in high school must contend with unintended biases on the part of teachers, counselors, and parents, as well as sometimes overt prejudices. The NCWIT Aspirations in Computing program was developed to provide these girls with encouragement, a network of support, and examples of success—all factors that have been shown to influence females to choose to study or work in computing.

A sense of belonging, or a feeling of "fit," is important for interest and persistence.

We know that girls' interest in computing classes is influenced by whether or not they have friends in the class or if boys dominate the classes.[1] In one survey of girls, nearly half (47%) said they would feel uncomfortable being the only girl in a group or class.[3] The Aspirations program enables girls to feel less isolated in these situations.

A sense of belonging, or a feeling of "fit," is important for interest and persistence. Subtle cues like sexist posters or "geeky" paraphernalia in a computer lab can suggest to girls that they do not belong.[1] Less subtle messages such as teachers, counselors, or parents steering female students to non-technical classes tell girls that technology is not for them, even when they may have interest or aptitude. Worse, even well-meaning adults sometimes believe that males have a "natural" talent for computing compared to females.[1] We know this feeling of "fitting in" is a major factor for

National winners of the NCWIT Aspirations in Computing Award at the NCWIT Summit in May, 2013.

PHOTOGRAPH BY CHRISTOPHER CARRUTH/CMC IMAGES

females in choosing a major and a career.[1] Programs like Aspirations can help to inoculate girls against feeling like a misfit.

Role models also can influence girls' decisions to pursue computing.[1] Such role models, however, are often less available for those students who do not come from affluent communities, positions of privilege, or school systems that provide high access to computing courses.[1] Some Aspirations awardees fall into these categories: In 2013, for example, 10% were from schools with 40% or more free or reduced lunch, 61% reported being a racial/ethnic minority, and many came from schools with few or no rigorous computing classes. One of the most important characteristics of a role model is that girls perceive these role models as "relatable" and similar to themselves in important ways. Awardees have reported that the support they receive and the role models they see among Aspirations awardees and NCWIT staff are essential to their continued pursuit of computing.

A Solution: The NCWIT Aspirations Award

Since 2007, NCWIT has been supporting young women through the NCWIT Award for Aspirations in Computing. Awardees are selected for their computing and IT aptitude, leadership ability, academic history, and plans for post-secondary education. The NCWIT Aspirations Award has a national competition as well as local competitions in all 50 states, plus Washington, D.C., Puerto Rico, and the U.S. Virgin Islands.

As the reach and awareness of the program increase, so do the number of applications, with 54% more girls applying in the 2013 season compared to 2012. With the increase in the number of applicants and regional awards, the number of winners and runners-up has also increased (see Figure 1).

The high school girls being rec-

Figure 1. 2007–2013 Aspirations Awards.

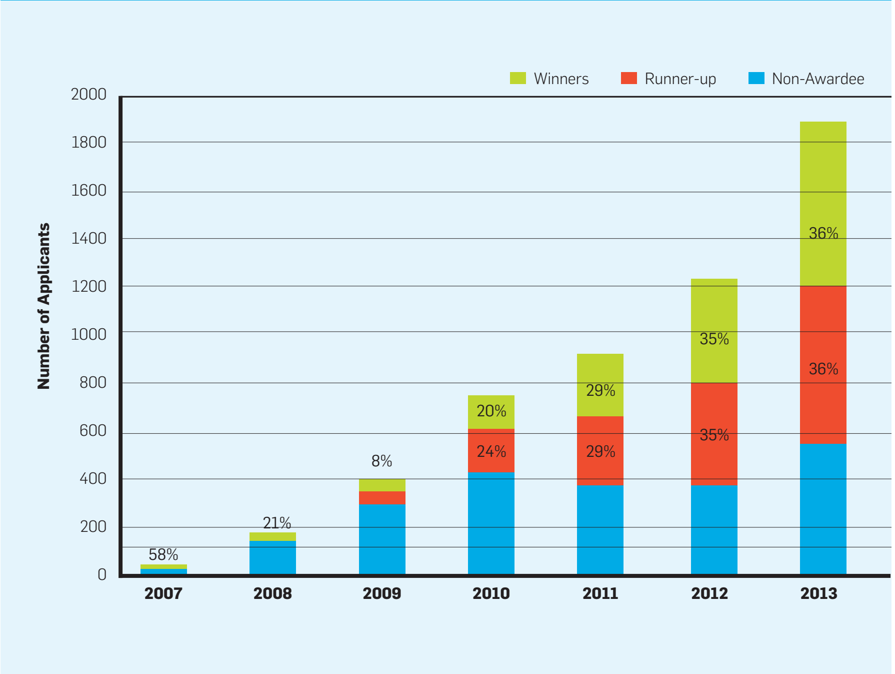
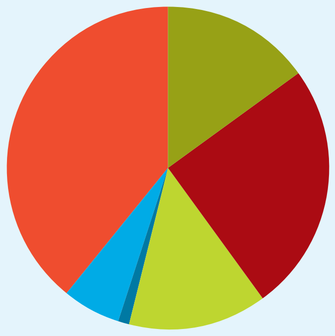


Figure 2. Racial/Ethnic percentages, on average, of the 2008–2013 Aspirations Award recipients.



ognized are a racially and ethnically diverse group, compared to the typical post-secondary computing department's student body. Figure 2 shows the racial/ethnic composition of the most recent season's awardees. Among awardees, 0.7% (n=8) identified as persons with disabilities.

Aspirations Affiliate Awards. NCWIT provides an online toolkit and other support to enable this program to be replicable and adaptable across the U.S. Typically, regional award events involve collaboration between community organizations, and academic and corporate entities. To date, 1,400 individuals from academia, non-profit organizations, and corporations have reviewed applications. Reviewers often remark on the high quality of the applications they read, which reinforces their desire to keep these talented young women in the field. The girls' applications are often truly inspirational, as evidenced by this excerpt: "I am most proud of starting a non-profit organization called Robot Springboard with my sister. Robot Springboard helps start robotics programs in underserved or underprivileged communities. This summer, my sister and I are teaching robotics camps in Kodiak, Alaska, a remote island community, and Wonder Park, Alaska."

Aspirations Award Program Outcomes. Evaluation data for the Aspirations Award program suggests the program is working. Awardees have reported greater confidence in their technical abilities, increased enthusiasm about computing, and greater awareness of the career opportunities available to them. Although 45% of Aspirations participants report not having taken a CS Advanced Placement exam, 71% of Aspirations participants in college are now majoring in a computer science or engineering field.

In the past award season, more than $300,000 in scholarships were offered by 45 NCWIT Academic Alliance institutions. Other opportunities for awardees include internships at NCWIT Workforce Alliance companies, complimentary registrations for technical conferences, meet-ups with corporate executives, and special events just for Aspirations recipients, such as the Apple TECHnically Women event and the Aspirations Day at Google.

The Aspirations in Computing Program is currently being studied by NCWIT research scientists DuBow and Wu. We have undertaken a three-year longitudinal, mixed-methods study assessing the influence of the Award for Aspirations in Computing on awardees' and non-awardees' choices of college major and career. We also will be studying the obstacles girls face in pursuing computing, and the circumstances under which they persist.

Another Solution: An Expanded Aspirations Program

Building off of NCWIT's Aspirations Award, the Aspirations in Computing program now spans from middle school, through high school and college, into the workforce.

Aspirations Educator Award. The NCWIT Aspirations in Computing Educator Award recognizes educators for their efforts to promote gender equity in computing. In the past three years, over 100 educators have been recognized and awarded over $100,000 in professional development funding.

One educator said: "As a result of having won the NCWIT Aspirations in Computing Educator Award, and

An image from the NCWIT photostream.

PHOTOGRAPH BY CHRISTOPHER CARRUTH/CMC IMAGES given my involvement in the Puerto Rico Institute of Robotics program as a mentor/teacher, I was recommended to participate in the NASA Summer of Innovation....For this Puerto Rican teacher—the daughter of poor people and the only one in my family to finish a college degree—it was overwhelming.... NCWIT opened the door for me to walk into a field full of new adventures and ideas." —Claribel Perez, Puerto Rico

Educator Award recipients have reported heightened interest in computing at their high schools and increased class enrollments (both female and male).

Middle School Program. In 2013, NCWIT launched AspireIT, a pilot initiative that matches Aspirations Award recipients with NCWIT Academic Alliance or NCWIT K–12 Alliance members to create and run computing-related outreach for middle school girls, such as after-school programs, summer camps, clubs, or weekend conferences. AspireIT employs a near-peer approach that provides middle school girls with a positive, inspiring experience of doing computing alongside their near peers from high school and college.

The AspireIT program has awarded $102,491 to 24 pilot programs in its inaugural round. These programs will run in 15 states, reaching a diverse 800 middle school girls with 25,000 hours of instruction. Preliminary evaluation findings suggest this program not only positively impacts the middle school student participants but also the Aspirations awardees leading the activities.

Aspirations Community. NCWIT established a restricted Facebook group for awardees (and more recently, one for all applicants) where the high school and college "members" can see and comment on scholarship and internship opportunities, as well as programming contests and other activities NCWIT posts. The Facebook group has become a place where the young women also discuss their own computing projects, college visits, job search and interview strategies, and how it feels to be female in a male-dominated field. The degree of support and information the Facebook group members offer one another is heartwarming, and often leads to in-person meet-ups at conferences and college campuses.

The Aspirations Program has the potential to bring many new women into the pipeline and retain thousands more.

In interviews and in Facebook postings, awardees frequently mention this cohort effect and its role in encouraging them to persist in the field and to not feel as isolated as they had in their local communities. One awardee noted in a survey, "Because of this award, I am less ashamed of who I am. I was known as a tomboy for such an immense interest in computers and I got embarrassed. Now, I'm proud!" The strength and influence of this peer community is an unintended outcome of the Aspirations program

Their postings suggest how much these young women yearn for encouragement and peer support. In this excerpt, a new awardee is introducing herself to the Facebook community: "Hi! ...I spend my free time doing Project Euler problems (projecteuler.net, really fun CS problems to do!). I really enjoy the NCWIT community! It's so inspiring to see so many women doing awesome things in technology, and I hope to be as cool as all of you one day.... Because of [the] NCWIT [award], I'm not afraid to try new things and I'm excited for a future in CS. Thank you all for being here and being so supportive!"

Get Involved

Through funding from multiple corporate partners and the National Science Foundation, the Aspirations Program has the potential to bring many new women into the pipeline and retain thousands more. Consider getting involved as a volunteer in this rewarding work. Both women and men are encouraged to volunteer; it takes all of us to make change.

- *Application Reviewer.* Help select the winners from thousands of applications submitted each year. Read and score online at your own pace during three weeks in November. Reviewers typically spend two to five hours per competition; you can choose to review applications from your own community or nationwide. Visit http://www.ncwit.org/review to sign up.
- *Committee Member.* Connect with an established award program in your community. Local Affiliate Award programs are in need of committee members for publicity, judging, and event planning.
- *AspireIT Host.* The AspireIT middle-school outreach program matches Aspirations award recipients with NCWIT Academic Alliance or K–12 Alliance members to create and run computing programs. If your organization is an NCWIT member, we can connect you with an award winner who wants to run a middle-school outreach program for girls.

For more details about the NCWIT Aspirations in Computing Program, see http://www.aspirations.org.

References

1. Ashcraft, C., Eger, E., and Friend, M. *Girls in IT: The Facts.* National Center for Women & Information Technology, 2012; http://www.ncwit.org/thefactsgirls.
2. Bandura, A. *Self-Efficacy: The Exercise of Control.* W.H. Freeman, NY, 1997
3. Girl Scout Research Institute. Generation STEM: What Girls Say about Science, Technology, Engineering, and Math; http://www.girlscouts.org/research/publications/stem/generation_stem_what_girls_say.asp

Wendy M. DuBow (wendy.dubow@colorado.edu) is a research scientist and the director of evaluation at the National Center for Women & Information Technology, University of Colorado-Boulder.

Ruthe Farmer (ruthe.farmer@colorado.edu) is the director of strategic initiatives at the National Center for Women & Information Technology, University of Colorado-Boulder.

Zhen Wu (zhen.wu@colorado.edu) is a research scientist at the National Center for Women & Information Technology, University of Colorado-Boulder.

Malia Fredrickson (malia.fredrickson@colorado.edu) is the Aspirations Award program manager at the National Center for Women & Information Technology, University of Colorado-Boulder.

The authors thank the U.S. National Science Foundation for support under award CNS-0813956, NCWIT's Strategic Partner Microsoft, and Investment Partners Avaya, Pfizer, and Bank of America. All opinions reflected in this column are those of the authors and not necessarily those of the U.S. National Science Foundation.

DOI:10.1145/2535918

Armando Fox

Viewpoint
From MOOCs to SPOCs

Supplementing the classroom experience with small private online courses.

IMAGE COLLAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

As the media's infatuation with massive open online courses (MOOCs) continues unabated, some academics seem to be succumbing to the hand-wringing about whether MOOCs will destroy higher education as we know it (see "Will MOOCs Destroy Academia?" by Moshe Vardi in the November 2012 issue of *Communications*). Is it a bad thing that we "have let the genie out of the bottle," as Vardi suggested in his Editor's Letter? I argue that a close, systematic, and sustained look at how MOOCs are actually being used should persuade the careful observer that tasteful use of MOOC technology can strengthen academia.

Note I do not say "MOOCs *will* strengthen academia." They certainly *can*, but whether they do depends on how they are received and used by academics. Full disclosure: besides being a MOOC instructor myself, I am the recently appointed faculty director of Berkeley's MOOCLab, which extends Berkeley's existing online education programs with MOOC research and practice. But I am not cheering for MOOCs because I have this position; rather, I agreed to take the position because I am excited about the possibilities of MOOCs and other online education. In particular, if MOOCs are used as a supplement to classroom teaching rather than being viewed a replacement for it, they can increase instructor leverage, student throughput, student mastery, and student engagement. I call this model the SPOC: small private online course.

To set the context for this discussion, let me use the SPOC idea to offer counterexamples to some "MOOC myths" in recent media coverage. While most myths are based on a kernel of truth and may be true of at least some MOOCs, they are just as often untrue and it is a disservice to interested readers to present them as foregone conclusions.

Myth: Universities will use MOOCs to lower costs by firing faculty and teaching assistants, thus sacrificing educational quality. If universities were looking to replace existing courses partially or entirely with MOOCs, this might be true. However, many universities are successfully using MOOC technology quite differently. For example, in a recent pilot program at San José State University in California, students in an analog circuits course used MIT-authored MOOC lectures and homework assignments created by Anant Agarwal.[1] The students' in-classroom time was spent working on lab and design problems with local faculty and teaching assistants. The students in this SPOC scored five percentage points higher on the first exam and 10 points on the second exam than the previous cohort that had used the traditional material. Even more strikingly, the proportion of students receiving credit for the course ("C" or better grade) increased from 59% to 91%. So educational quality arguably increased, and costs were lowered by helping students graduate more quickly, rather than by firing people. Productivity was enhanced because the on-campus instructors shifted their time from what they perceived as a lower-value activity—creating and delivering lectures on content that has not changed much—to the higher-value activity of working directly with students on the material. Several of my colleagues in the California State University system and the community college system have expressed similar enthusiasm. This model takes advantage of important MOOC features, including access to high-quality materials and rapid feedback to students via autograding, to maximize the leverage of the scarce resource—instructor time.

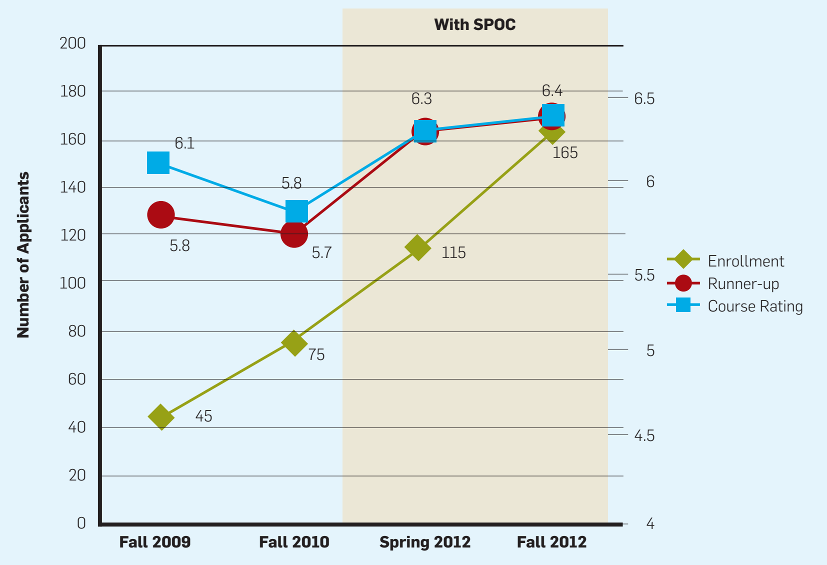
Closer to home, my colleague David Patterson and I created a MOOC based on our upper-division software engineering course at Berkeley, and subsequently used the MOOC material as a SPOC in our on-campus course. A key feature of this course is four different autograders for different types of software engineering assignments. These autograders were created by investing several hundred engineer-hours in repurposing tools used by professional programmers. Students not only get finer-grained feedback than they would get from human teaching assistants, who can spend at most a few minutes per assignment, but now have the opportunity to resubmit homework to improve on their previous score and increase mastery. The autograders test both code completeness and code correctness, and will soon give feedback on code style. As the accompanying figure shows, the SPOC model has allowed us to increase the enrollment of the course nearly fourfold while yielding higher instructor and course ratings (in fact, the highest in the course's 20-year history) even though the fundamental material covered has changed very little. (The MOOC version of the course is available as "BerkeleyX CS169.1x" on edx.org.)

Myth: MOOCs will fail because many aspects of traditional classes, such as small-group discussions and face-to-face time with instructors, do not work in the MOOC format. This assertion is true, but it implicitly and incorrectly assumes that replicating the classroom experience is the proper goal for an online course. If that were an appropriate goal, then MOOCs would indeed fail to meet it. However, as educators, a better question for us to ask is this: What can be delivered effectively through this medium in a way that helps our on-campus students, and has the valuable side effect of helping the hundreds of thousands who will not have the privilege of attending our universities in person? (Indeed, many of our MOOC students reported that our course was better than anything available at the brick-and-mortar campuses to which they had access.) Using MOOC materials in a SPOC format is one way that MOOCs can indeed be successful in helping to answer this broader question.

Under what conditions and with what types of material do online communities foster learning?

For example, rather than asking whether automatic graders (which, by the way, have been around since at least 1960[4]) can replace individual instructor attention, we can ask: When can they relieve teaching staff of drudgery, allowing scarce instructor time to focus on higher-value interactions such as tutoring and design reviews? Rather than worrying whether MOOC-based social networking will replace face-to-face peer interactions, we can ask and experimentally answer: Under what conditions and with what types of material do online communities help foster learning, and how can social networking technology help foster both online and in-person community building? And learning activities that do not appear to be "MOOCable"—discussion-based learning, open-ended design projects, and so on—can just be omitted from the MOOC but covered in the classroom setting, as we have done in our software engineering course, whose MOOC version lacks the on-campus

Course enrollment and instructor and course ratings (given anonymously by enrolled students, solicited by Eta Kappa Nu Engineering Honor Society each semester within Berkeley Engineering) of CS 169 Software Engineering with and without SPOC supplement.



course's open-ended design project. Indeed, at universities on the quarter system, it is common to offer a two-quarter sequence in which the first quarter focuses on well-circumscribed assignments and the second quarter focuses on a design project, since a single quarter cannot cover both. The first course clearly has value despite lacking a design project, and could be offered as either a MOOC or a SPOC. By analogy, MOOCs that do not offer "the same" experience as a complete residential course also have value, and our job as educators is to make judgments about where that value lies and how to combine it with the other education modalities we offer our students.

Myth: MOOCs distract faculty who should be focusing on improving their on-campus pedagogy. Even if using a SPOC in the classroom, faculty can still leverage the scale of an (open) MOOC to enhance their classroom teaching. In fact, the large enrollments of MOOCs offer us new and unprecedented opportunities to improve our on-campus courses using inferential statistics techniques that just do not work at smaller scales, and so were previously available only to large-enrollment "high stakes" exams such as the GRE or SAT.[a] For example, exploratory factor analysis[5] lets us identify questions that test comparable concepts, giving instructors a way to vary exam content. Item response theory[6] allows us to discover which questions are more difficult (in the statistical sense that higher-performing students are more likely to get them right). A/B testing gives us a controlled way to evaluate which approaches have better effects on learning outcomes, just as high-volume e-commerce sites evaluate which user experience results in more purchases. None of these techniques works on classroom-sized cohorts (say, 200 or fewer students), but we are applying all of them to our current MOOC. Indeed, not all instructors will be eager to receive the avalanche of MOOC

a The Graduate Record Exam (GRE) and Scholastic Aptitude Test (SAT) are standardized tests that are part of most students' applications to U.S. graduate and undergraduate programs respectively.

Both MOOCs and SPOCs are two design points in a wider space in which experiments are possible.

data telling us what is not working in our courses and how we can improve them, but our sense at Berkeley is that MOOCs may well raise the bar for acceptable teaching on campus, as well as improve the recognition of good teaching, perhaps finally bringing to a close the era of recycled PowerPoint slides.

In addition, in each of four offerings of our software engineering MOOC totaling over 100,000 enrollees, about 8%, or nearly 32,000 total, identified themselves as instructors, suggesting that MOOCs may be even more effective than traditional textbooks at "teaching the teachers" and getting innovative new pedagogy out to a large audience. In fact, our faculty colleagues who are classroom-testing our unconventional new textbook *Engineering Long-Lasting Software: An Agile Approach Using SaaS & Cloud Computing* are all doing so in conjunction with our MOOC (EdX CS 169.1x), so they can take advantage of the autograders, screencasts, and other materials.

Myth: MOOCs will reduce diversity in instructors and teaching approaches because economics will favor a "winner takes all" scenario in which one specific MOOC will dominate each course. In her widely cited *Tools For Teaching*,[2] Davis recommends that lecture styles and teaching strategies should vary depending on the nature of the material and the target audience of students. Even if one or a few MOOCs dominate a particular course, thereby replacing various instructors' different teaching approaches with the MOOC instructor's single approach, we can, like Doug Fisher and others,[3] selectively adapt the content for SPOC use in our own on-campus courses, as we have long done with textbooks. Indeed, one could have raised a similar complaint about the printing press: it homogenized book production and eliminated the social rituals associated with acquiring books. Yet it also created vastly more readers, gave voices to authors who would never have had them, and introduced new tools that teachers could use in conjunction with their lecturing. In a similar way, MOOCs will not replace high-quality face-to-face instruction, but we can reach many more learners, leading to a net social and economic benefit, and we can give many great teachers a more prominent voice than they have had since Socrates.

Conclusion

MOOCs represent a new technology opportunity whose potential pedagogical impact needs to be researched. I have argued that MOOCs themselves can yield valuable information because of their scale, and that MOOC materials can be used in a blended small private online course setting to supplement the classroom experience. Both MOOCs and SPOCs are two design points in a wider space in which experiments are possible. To be sure, many bad experiments will be tried—some are probably already under way—and many worthy experiments will fail or have a different outcome than desired. But if failed experiments were an obstacle to doing world-changing research, we academics would probably choose a different job.

References

1 California to give web courses a big trial. *The New York Times* (Jan. 5, 2013); http://www.nytimes.com/2013/01/15/technology/california-to-give-web-courses-a-bigtrial.html?_r=0.
2. Davis, B.G. *Tools for Teaching*. Jossey-Bass, 2009.
3. Fisher, D. Warming up to MOOCs. Chronicle of Higher Education (Nov. 6, 2012); http://chronicle.com/blogs/profhacker/warming-up-to-moocs/44022.
4. Hollingsworth, J. Automatic graders for programming classes. *Commun. ACM 3*, 10 (Oct. 1960).
5. Lawley, D. Estimation of factor loadings by the method of maximum likelihood. In *Proceedings of the Royal Society of Edinburgh*, 60A, 1940.
6. Lord, F.M. *Applications of Item Response Theory to Practical Testing Problems*. Erlbaum, Mahwah, NJ, 1980.

Armando Fox (fox@cs.berkeley.edu) is Professor in Residence at UC Berkeley and a co-founder of the Berkeley RAD Lab.

HEIDELBERG LAUREATE FORUM

acm Association for Computing Machinery

Inviting Young Scientists

Meet Some of the Greatest Minds of Mathematics and Computer Science

Young researchers in the fields of mathematics and/or computer science are invited to participate in an extraordinary opportunity to meet some of the preeminent scientists in the field. ACM has joined forces with the Heidelberg Laureate Forum (HLF) to bring students together with the very pioneering researchers who may have sparked their passion for science and math. These role models include recipients of the Abel Prize, the ACM A.M. Turing Award, and the Fields Medal.

The next Heidelberg Laureate Forum will take place September 21–26, 2014 in Heidelberg, Germany.

The week-long event will focus on scientific inspiration and exchange through a series of presentations, workshops, panel discussions, and social events involving both the laureates and the young scientists.

Who can participate?

The HLF invites new and recent Ph.D's, Ph.D. candidates, other graduate students involved in research and undergraduate students with solid experience in and a commitment to computing research to apply.

How to apply:

Young researchers can apply online:

https://application.heidelberg-laureate-forum.org/

The materials required for a complete application are listed on the site.

What is the schedule?

The deadline for applications is **February 15, 2014**.

We reserve the right to close the application website early should we receive more applications and nominations than our reviewers can handle.

Successful applicants will be notified by **April 15, 2014** and will receive full support to attend the Forum.

practice

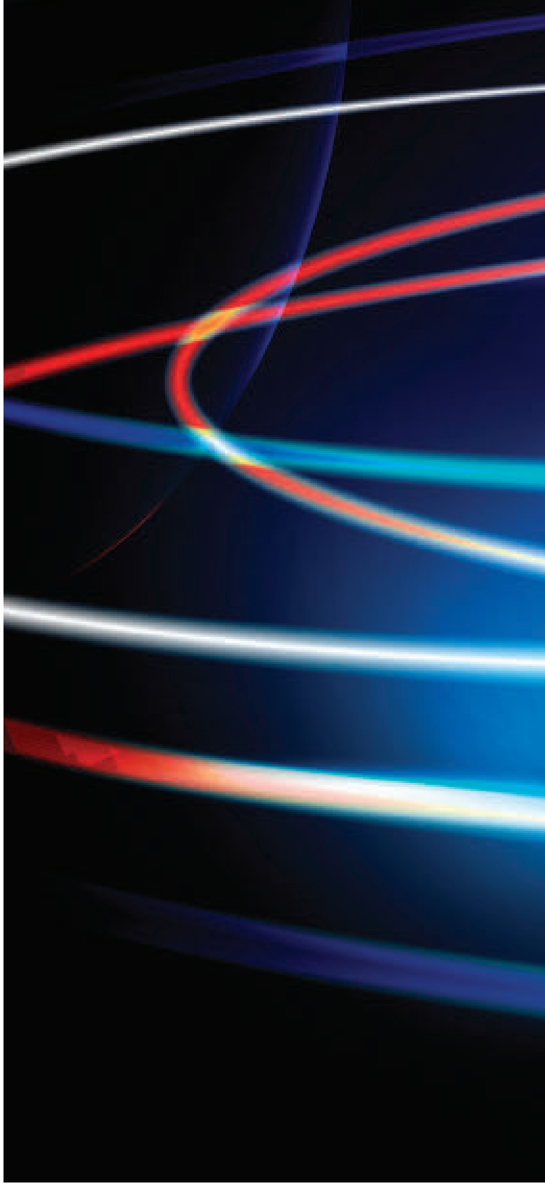
DOI:10.1145/2534706.2534721

Article development led by acmqueue
queue.acm.org

HTTP continues to evolve.

BY ILYA GRIGORIK

Making the Web Faster with HTTP 2.0



HYPERTEXT TRANSFER PROTOCOL (HTTP) is one of the most widely used application protocols on the Internet. Since its publication, RFC 2616 (HTTP 1.1) has served as a foundation for the unprecedented growth of the Internet: billions of devices of all shapes and sizes, from desktop computers to the tiny Web devices in our pockets, speak HTTP every day to deliver news, video, and millions of other Web applications we have all come to depend on in our everyday lives.

What began as a simple one-line protocol for retrieving hypertext (that is, `"GET/document"`) quickly evolved into a generic hypermedia transport. Now a decade later it is used to power just about any use case imaginable.

Under the weight of its own success, however, and as more and more everyday interactions continue to migrate to the Web—social, email, news and video, and, increasingly, our personal and job workspaces—HTTP has begun to show signs of stress. Users and developers alike are now demanding near-real-time responsiveness and protocol performance from HTTP 1.1, which it simply cannot meet without some modifications.

To meet these new challenges, HTTP must continue to evolve, which is where HTTP 2.0 enters the picture. HTTP 2.0 will make applications faster, simpler, and more robust by enabling efficient multiplexing and low-latency delivery over a single connection and allowing Web developers to undo many of the application "hacks" used today to work around the limitations of HTTP 1.1.

Performance Challenges of Modern Web Applications

A lot has changed in the decade since the HTTP 1.1 RFC was published: browsers have continued to evolve at an accelerating rate, user connectivity profiles have changed with the mo-

ILLUSTRATION BY IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES

bile Web now at an inflection point, and Web applications have grown in their scope, ambition, and complexity. Some of these factors help performance while others hinder. On balance, Web performance remains a large and unsolved problem.

First, the good news: modern browsers have put significant effort into performance. JavaScript execution speed continues its steady climb (for example, the launch of the Chrome browser in 2008 delivered a 20x improvement, and in 2012 alone, the performance was further improved by more than 50% on mobile[10]). And it is not just JavaScript where the improvement is occurring; modern browsers also leverage GPU acceleration for drawing and animation (for example, CSS3 animations and WebGL), provide direct access to native device APIs, and leverage numerous speculative optimization techniques[4] to help hide and reduce various sources of network latency.

Similarly, broadband adoption (see Table 1) has continued its steady climb over the past decade. According to Akamai, while the global average is now at 3.1Mbps, many users have access to far higher throughput, especially with the rollout of residential fiber solutions.[1] Bandwidth is only half the equation, however. Latency is the oft-forgotten factor, and unfortunately, it is now often the *limiting factor* when it comes to browsing the Web.[3]

In practice, once the user has more than 5Mbps of bandwidth, further improvements deliver minimal increase in the loading speed of the average Web application: streaming HD video from the Web is bandwidth bound; loading the page hosting the HD video, with all of its assets, is latency bound.

A modern Web application looks significantly different from a decade ago. According to HTTP Archive,[5] an average Web application is now composed of more than 90 resources, which are fetched from more than 15 distinct hosts, totaling more than 1,300KB of (compressed) transferred data. As a result, a large fraction of HTTP data flows consist of small (less than 15KB), bursty data transfers over dozens of distinct TCP connections. Therein lies the problem. TCP is optimized for long-lived connections and bulk data transfers. Network RTT (round-trip time) is the limiting factor in throughput of new TCP connections (a result of TCP congestion control), and consequently, latency is also the performance bottleneck for most Web applications.

How do you address this mismatch? First, you could try to reduce the round-trip latency by positioning the servers and bits closer to the user, as well as using lower-latency links. Unfortunately, while these are necessary optimizations—there is now an entire content delivery network (CDN) industry focused on exactly this problem—

they are not sufficient. As an example, the global average RTT to google.com, which employs all of these techniques, was approximately 100 milliseconds in 2012, and unfortunately this number has not budged in the past few years.

Many existing links are already within a small constant factor (1.2˜1.5) of the speed-of-light limit, and while there is still room for improvement, especially with respect to "last-mile latency," the relative gains are modest. Worse, with the rise of mobile networks, the impact of the latency bottleneck has only gotten worse. While the latest 4G mobile networks are specifically targeting low-latency data delivery, the advertised and real-world performance is still often measured in hundreds of milliseconds of overhead (see Table 2 for advertised latencies in the AT&T core radio networks[2]).

If you cannot get the performance-step function needed from improving the underlying links—if anything, with the rise of mobile traffic, there is a regression—then you must turn your attention to how you construct applications and tune the performance of underlying transport protocols responsible for their delivery.

Performance Limitations of HTTP 1.1

Improving the performance of HTTP was one of the key design goals for the HTTP 1.1 Working Group, and the standard introduced many critical performance enhancements. A few of the best known include:

- Persistent connections to allow connection reuse.
- Chunked transfer encoding to allow response streaming.
- Request pipelining to allow parallel request processing.
- Byte serving to allow range-based resource requests.
- Improved and much better specified caching mechanisms.

Unfortunately, some HTTP 1.1 features such as request pipelining have effectively failed due to lack of support and deployment challenges; while some browsers today support pipelining as an optional feature, few if any have it enabled by default. As a result, HTTP 1.1 forces strict request queuing on the client (Figure 1): the client dispatches the request and must wait until the response is returned by the server, which means a single large transfer or a slow dynamic resource can block the entire connection. Worse, the browser has no way of reliably predicting this behavior and, as result, is often forced to rely on heuristics to guess whether it should wait and attempt to reuse the existing connection or open another one.

In light of the limitations of HTTP 1.1, the Web developer community—always an inventive lot—has created and popularized a number of home-brew application workarounds (calling them *optimizations* would give them too much credit):

- Modern browsers allow up to six parallel connections per origin, which effectively allows up to six parallel resource transfers. Not satisfied with the limit of six connections, many developers decided to apply *domain sharding*, which splits site resources across different origins, thereby allowing more TCP connections. Recall that an average page now talks to 15 distinct hosts, each of which might use as many as six TCP connections.
- Small files with the same file type are often concatenated together, creating larger bundles to minimize HTTP request overhead. In effect, this is a form of multiplexing, but it is applied at the application layer—for example, Cascading Style Sheets (CSS) and JavaScript files are combined into larger bundles, small images are merged into image sprites, and so on.
- Some files are inlined directly into the HTML document to avoid the HTTP request entirely.

For many Web developers all of these are matter-of-fact optimizations—familiar, necessary, and universally accepted. Each of these workarounds, however, also often carries many negative implications both for the complexity and the performance of applications:

- Aggressive sharding often causes network congestion and is counterproductive, leading to: additional and unnecessary DNS (Domain Name Service) lookups and TCP handshakes; higher resource load caused by more sockets on client, server, and intermediaries; more network contention between parallel streams; and so on.
- Concatenation breaks the modularity of application code and has a negative impact on caching (for example, a common practice is to concatenate all JavaScript or CSS files into large bundles, which forces download and invalidation of the entire bundle on a single byte change). Similarly, JavaScript and CSS files are parsed and executed only when the entire file is downloaded, which adds processing delays; large image sprites also occupy more memory on the client and require more resources to decode and process.

Table 1. Global broadband adoption.

Rank	Country	Average Mbps	Year Over Year Change
—	Global	3.1	17%
1	South Korea	14.2	–10%
2	Japan	11.7	6.8%
3	Hong Kong	10.9	16%
4	Switzerland	10.1	24%
5	Netherlands	9.9	12%
…			
9	United States	8.6	27%

Table 2. Advertised latencies.

	LTE	HSPA+	HSPA	EDGE	GPRS
Latency	40–50 ms	100–200 ms	150–400 ms	600–750 ms	600–750 ms

- Inlined assets cannot be cached individually and inflate the parent document. A common practice of inlining small images also inflates their size by more than 30% via base64 encoding and breaks request prioritization in the browser—typically, images are fetched with lower priority by the browser to accelerate page construction.

In short, many of the workarounds have serious negative performance implications. Web developers should not have to worry about concatenating files, spiriting images, inlining assets, or domain sharding. All of these techniques are stopgap workarounds for limitations of the HTTP 1.1 protocol. Hence, HTTP 2.0.

HTTP 2.0 Design and Technical Goals

Developing a major revision of a protocol underlying all Web communication is a nontrivial task requiring a lot of careful thought, experimentation, and coordination. As such, it is important to define a clear technical charter and, arguably even more importantly, define the boundaries of the project. The intent is not to overhaul every detail of the protocol but to make meaningful though incremental progress to improve Web performance.

With that, the HTTPbis Working Group charter[6] for HTTP 2.0 is scoped as follows:

- Substantially and measurably improve end-user-perceived latency in most cases over HTTP 1.1 using TCP.
- Address the HOL (head-of-line) blocking problem in HTTP.
- Do not require multiple connections to a server to enable parallelism, thus improving its use of TCP, especially regarding congestion control.
- Retain the semantics of HTTP 1.1, leveraging existing documentation, including (but not limited to) HTTP methods, status codes, URIs, and where appropriate, header fields.
- Clearly define how HTTP 2.0 interacts with HTTP 1.x, especially in intermediaries.
- Clearly identify any new extensibility points and policy for their appropriate use.

To deliver on these goals HTTP 2.0 introduces a new layering mechanism onto TCP, which addresses the well-known performance limitations of HTTP 1.x. The application semantics of HTTP remain untouched, and no changes are being made to the core concepts such as HTTP methods, status codes, URIs, and header fields—these changes are explicitly out of scope. With that in mind, let's take a look "under the hood" of HTTP 2.0.

Request and response multiplexing. At the core of all HTTP 2.0's performance enhancements is the new binary framing layer (see Figure 2), which dictates how HTTP messages are encapsulated and transferred between the client and server. HTTP semantics such as verbs, methods, and headers are unaffected, but the way they are encoded while in transit is different.

With HTTP 1.x, if the client wants to make multiple parallel requests to improve performance, then multiple TCP connections are required. This behavior is a direct consequence of the newline-delimited plaintext HTTP 1.x protocol, which ensures only one response at a time can be delivered per connection—worse, this also results in HOL blocking and inefficient use of the underlying TCP connection.

The new binary framing layer in HTTP 2.0 removes these limitations and enables full request and response multiplexing. The following HTTP 2.0

Figure 1. With 56ms RTT, fetching two files takes approximately 228ms, with 80% of that time in network latency.

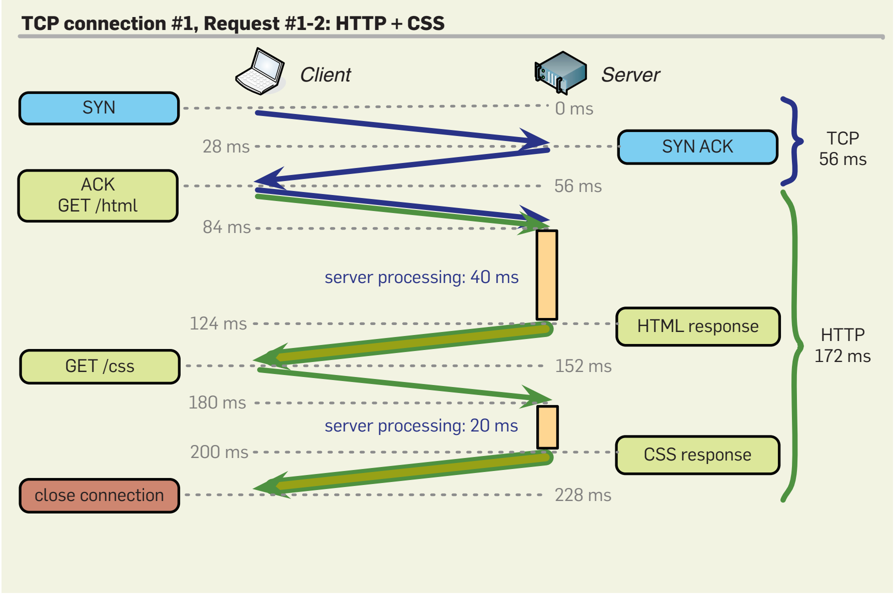
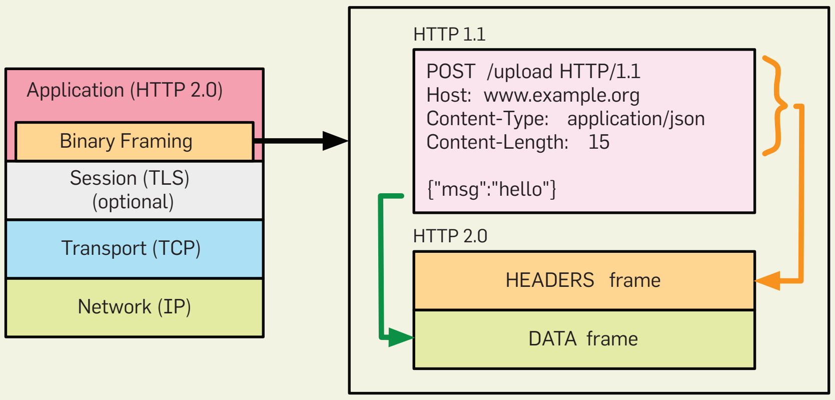


Figure 2. HTTP 2.0 binary framing.



terminology will help in understanding this process:

▸ *Stream*—a bidirectional flow of bytes, or a virtual channel, within a connection. Each stream has a relative priority value and a unique integer identifier.

▸ *Message*—a complete sequence of frames that maps to a logical message such as an HTTP request or a response.

▸ *Frame*—the smallest unit of communication in HTTP 2.0, each containing a consistent frame header, which at minimum identifies the stream to which the frame belongs, and carries a specific type of data (for example, HTTP headers, payload, and so on).

All HTTP 2.0 communication can be performed within a single connection that can carry any number of bi directional *streams*. In turn, each stream communicates in *messages*, which consist of one or multiple *frames*, each of which may be interleaved (see Figure 3) and then reassembled via the embedded stream identifier in the header of each individual frame.

The ability to break down an HTTP message into independent frames, prioritize and interleave them within a shared connection, and then reassemble them on the other end is the single most important enhancement of HTTP 2.0. By itself, this change is entirely unremarkable, since many protocols below HTTP already implement similar mechanisms. This "small" change, however, introduces a ripple effect of numerous performance benefits across the entire stack of all Web technologies, allowing developers to do the following:

▸ Interleave multiple requests in parallel without blocking on any one.

▸ Interleave multiple responses in parallel without blocking on any one.

▸ Use a single connection to deliver many requests and responses in parallel.

▸ Deliver lower page-load times by eliminating unnecessary latency.

▸ Remove unnecessary HTTP 1.x workarounds from application code.

▸ And much more…

Binary framing. HTTP 2.0 uses a binary, length-prefixed framing layer, which offers more compact representation than the newline-delimited plaintext HTTP 1.x protocol and is both easier and more efficient to process. All HTTP 2.0 frames share a common eight-byte header (see Figure 4), which contains the length of the frame, its type, a bit field for flags, and a 31-bit stream identifier.

▸ The 16-bit length prefix reveals a single frame can carry $2^{16}–1$ bytes of data—˜64KB—which excludes the 8-byte header size.

▸ The 8-bit type field determines how the rest of the frame is interpreted.

▸ The 8-bit flags field allows different frame types to define frame-specific messaging flags.

▸ A 1-bit reserved field is always set to 0.

▸ The 31-bit stream identifier uniquely identifies the HTTP 2.0 stream.

Given this knowledge of the shared HTTP 2.0 frame header, you can write a simple parser that can examine any HTTP 2.0 bytestream, identify different frame types, and report their flags and the length of each by examining the first eight bytes of every frame. Further, because each frame is length prefixed, the parser can skip ahead to the beginning of the next frame quickly and efficiently. This is a big pevrformance improvement over HTTP 1.x.

Once the frame type is known, the

Figure 3. Interleaved frames from multiple in-flight HTTP 2.0 streams within a single connection.

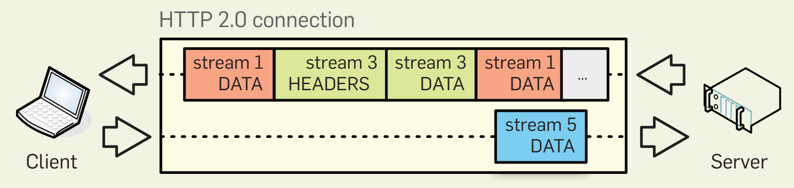


Figure 4. Common eight-byte frame header.

Bit	+0..7	+8..15	+16..23	+24..31
0	Length		Type	Flags
32	R	Stream Identifier		
…	*Frame Payload*			

Figure 5. HEADERS frame with stream priority and header payload.

Bit	+0..7	+8..15	+16..23	+24..31
0	Length		Type (1)	Flags
32	R	Stream Identifier		
64	R	Priority		
…	*Header Block*			

Figure 6. DATA frame.

Bit	+0..7	+8..15	+16..23	+24..31
0	Length		Type (0)	Flags
32	R	Stream Identifier		
…	*HTTP Payload*			

parser can interpret the remainder of the frame. The HTTP 2.0 standard defines the types listed in Table 3.

Full analysis of this taxonomy of frames is outside the scope of this article—after all, that is what the spec[7] is for (and it does a great job!). Having said that, let's go just one step further and look at the two most common workflows: initiating a new stream and exchanging application data.

Table 3. HTTP 2.0 frame types.

DATA	used to transport HTTP message bodies
HEADERS	used to communicate additional header fields for a stream
PRIORITY	used to assign or reassign priority of referenced resource
RST_STREAM	used to signal abnormal termination of a stream
SETTINGS	used to signal configuration data about how two endpoints may communicate
PUSH_PROMISE	used to signal a promise to create a stream and serve referenced resource
PING	used to measure the round-trip time and perform "liveness" checks
GOAWAY	used to inform the peer to stop creating streams for current connection
WINDOW_UPDATE	used to implement flow control on per-stream or per-connection basis
CONTINUATION	used to continue a sequence of header block fragments

Initiating New HTTP 2.0 Streams

Before any application data can be sent, a new stream must be created and the appropriate metadata such as HTTP headers must be sent. That is what the HEADERS frame is for (see Figure 5).

Notice that in addition to the common header, an optional 31-bit stream priority has been added. As a result, whenever the client initiates a new stream, it can signal to the server the relative priority of that request, and even reprioritize it later by sending another PRIORITY frame.

How do the client and server negotiate the unique stream IDs? They do not. Client-initiated streams have even-numbered stream IDs and server-initiated streams have odd-numbered stream IDs. This offset eliminates collisions in stream IDs between the client and server.

Finally, the HTTP header key-value pairs are encoded via a custom header compression algorithm (more on this later) to minimize the size of the payload, and they are appended to the end of the frame.

Notice the HEADERS frames are used to communicate only the metadata about each stream. The actual application payload is delivered independently within the frame payload (see Figure 6) that follow them (that is, there is a separation between "data" and "control" messaging).

The DATA frame is trivial: it is the common 8-byte header followed by the actual payload. To reduce HOL blocking, the HTTP 2.0 standard requires that each DATA frame not exceed $2^{14}-1$ (16,383) bytes, which means that larger messages have to be broken up into smaller chunks. The last message in a sequence sets the END_STREAM flag to mark the end of data transfer.

There are a few more implementation details, but this information is enough to build a very basic HTTP 2.0 parser—emphasis on *very basic*. Features such as stream prioritization, server push, header compression, and flow control (not yet mentioned) warrant a bit more discussion, as they are critical to getting the best performance out of HTTP 2.0.

Stream prioritization. Once an HTTP message can be split into many individual frames, the exact order in which the frames are interleaved and delivered within a connection can be optimized to further improve the performance of an application. Hence, the optional 31-bit priority value: 0 represents the highest-priority stream; $2^{31}-1$ represents the lowest-priority stream.

Not all resources have equal priority when rendering a page in the browser: the HTML document is, of course, critical, as it contains the structure and references to other resources; CSS is required to create the visual rendering tree (you cannot paint pixels until you have the style-sheet rules); increasingly, JavaScript is also required to bootstrap the page; remaining resources such as images can be fetched with lower priority.

The good news is that all modern browsers already perform this sort of internal optimization by prioritizing different resource requests based on type of asset, their location on the page, and even learned priority from previous visits[4] (for example, if the rendering was blocked on a certain asset in a previous visit, the same asset may be prioritized higher in the future).

With the introduction of explicit stream prioritization in HTTP 2.0 the browser can communicate these inferred priorities to the server to improve performance: the server can prioritize stream processing by controlling the allocation of resources (CPU, memory, bandwidth); and once the response data is available, the server can prioritize delivery of high-priority frames to the client. Even better, the client is now able to dispatch all of the requests as soon as they are discovered (that is, eliminate client-side request queueing latency) instead of relying on request prioritization heuristics in light of limited parallelism provided by HTTP 1.x.

Server push. A powerful new feature of HTTP 2.0 is the ability of the server to send multiple replies for a single client request—that is, in addition to the response to the original request, the server can push additional resources to the client without having the client explicitly request each one.

Why would such a mechanism be needed? A typical Web application consists of dozens of resources, all of which the client discovers by examining the document provided by the server. As a result, why not eliminate the extra latency and let the server push the associated resources to the client ahead of time? The server already knows which resources the client will require—that is *server push*.

In fact, while support for server push as an HTTP protocol feature is new, many Web applications are already using it, just under a different name: *inlining*. Whenever the developer inlines an asset—CSS, JavaScript, or any other asset via a data URI—they are, in effect, pushing that resource to the client instead of waiting for the client to request it. The only difference with

Figure 7. Differential encoding of HTTP 2.0 headers.

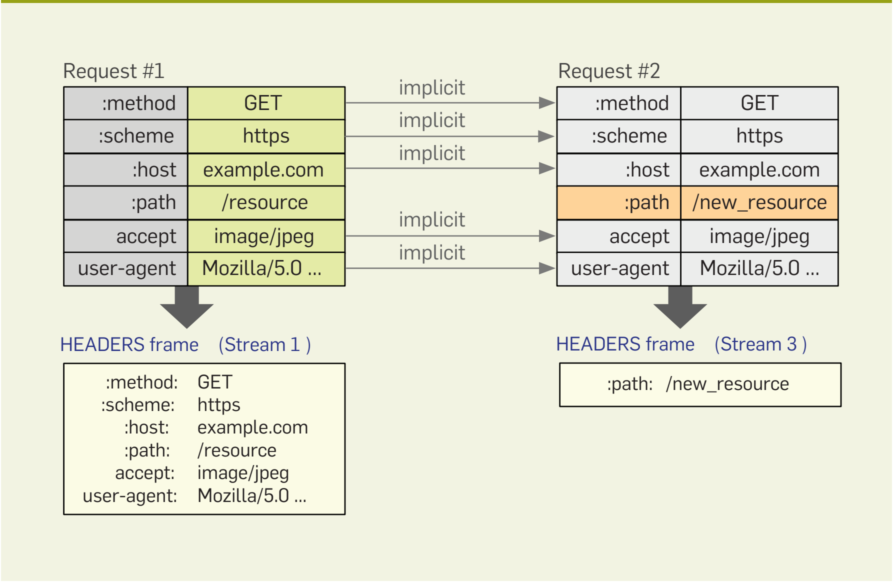


Figure 8. HTTP Upgrade mechanism.

```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: HTTP/2.0
HTTP2-Settings: (SETTINGS payload)

HTTP/1.1 200 OK
Content-length: 243
Content-type: text/html

(... HTTP 1.1 response ...)

          (or)

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: HTTP/2.0

(... HTTP 2.0 response ...)
```

HTTP 2.0 is that this workflow can now move out of the application and into the HTTP protocol itself, which offers important benefits: pushed resources can be cached by the client, declined by the client, reused across different pages, and prioritized by the server.

In effect, server push makes obsolete most of the cases where inlining is used in HTTP 1.x.

Header compression. Each HTTP transfer carries a set of headers used to describe the transferred resource. In HTTP 1.x, this metadata is always sent as plaintext and typically adds anywhere from 500 to 800 bytes of overhead per request, and often much more if HTTP cookies are required. To reduce this overhead and improve performance, HTTP 2.0 compresses header metadata:[8]

- Instead of retransmitting the same data on each request and response, HTTP 2.0 uses header tables on both the client and server to track and store previously sent header key-value pairs.
- Header tables persist for the entire HTTP 2.0 connection and are incrementally updated both by the client and server.
- Each new header key-value pair is either appended to the existing table or replaces a previous value in the table.

As a result, both sides of the HTTP 2.0 connection know which headers have been sent, and their previous values, which allows a new set of headers to be coded as a simple difference (see Figure 7) from the previous set.

Common key-value pairs that rarely change throughout the lifetime of a connection (for example, user-agent, accept header, and so on), need to be transmitted only once. In fact, if no headers change between requests (for example, a polling request for the same resource), then the header-encoding overhead is zero bytes—all headers are automatically inherited from the previous request.

Flow control. Multiplexing multiple streams over the same TCP connection introduces contention for shared bandwidth resources. Stream priorities can help determine the relative order of delivery, but priorities alone are insufficient to control how the resource allocation is performed between multiple streams. To address this, HTTP 2.0 provides a simple mechanism for stream and connection flow control:

- Flow control is hop-by-hop, not end-to-end.
- Flow control is based on window update frames: the receiver advertises how many bytes of DATA-frame payload it is prepared to receive on a stream and for the entire connection.
- Flow-control window size is updated via a WINDOW_UPDATE frame that specifies the stream ID and the window increment value.
- Flow control is directional—the receiver may choose to set any window size it desires for each stream and for the entire connection.
- Flow control can be disabled by a receiver.

As experience with TCP shows, flow control is both an art and a science. Research on better algorithms and implementation improvements are continuing to this day. With that in mind, HTTP 2.0 does not mandate any specific approach. Instead, it simply provides the necessary tools to implement such an algorithm—a great area for further research and optimization.

Efficient HTTP 2.0 upgrade and discovery. Though there are a lot more technical and implementation details, this whirlwind tour of HTTP 2.0 has covered the highlights: binary framing, multiplexing, prioritization, server push, header compression, and flow control. Combined, these features will deliver significant performance improvements on both the client and server.

Having said that, there is one more minor detail: how does one deploy a major revision of the HTTP protocol? The switch to HTTP 2.0 cannot happen overnight. Millions of servers must be updated to use the new binary framing protocol, and billions of clients must similarly update their browsers and networking libraries.

The good news is that most modern browsers use efficient background update mechanisms, which will enable HTTP 2.0 support quickly and with minimal intervention for a large portion of existing users. Despite this, some users will be stuck with older browsers, and servers and intermediaries will also have to be updated to support HTTP 2.0, which is a much longer labor- and capital-intensive process.

HTTP 1.x will be around for at least another decade, and most servers and clients will have to support both 1.x and 2.0 standards. As a result, an HTTP 2.0-capable client must be able to discover whether the server—and any and all intermediaries—support the HTTP 2.0 protocol when initiating a new HTTP session. There are two cases to consider:

- Initiating a new (secure) HTTPS connection via TLS.
- Initiating a new (unencrypted) HTTP connection.

In the case of a secure HTTPS connection, the new ALPN (Application Layer Protocol Negotiation[9]) extension to the TLS protocol allows users to negotiate HTTP 2.0 support as part of the regular TLS handshake: the client sends the list of protocols it supports (for example, http/2.0); the server selects one of the advertised protocols and confirms its choice by sending the protocol name back to the client as part of the regular TLS handshake.

Establishing an HTTP 2.0 connection over a regular, nonencrypted channel requires a bit more work. Because both HTTP 1.0 and HTTP 2.0 run on the same port (80), in the absence of any other information about the server's support for HTTP 2.0, the client will have to use the HTTP Upgrade mechanism to negotiate the appropriate protocol, as shown in Figure 8.

Using the Upgrade flow, if the server does not support HTTP 2.0, then it can immediately respond to the request with an HTTP 1.1 response. Alternatively, it can confirm the HTTP 2.0 upgrade by returning the "101 Switching Protocols" response in HTTP 1.1 format, and then immediately switch to HTTP 2.0 and return the response using the new binary framing protocol. In either case, no extra round-trips are incurred.

Crystal Gazing

Developing a major revision of a protocol underlying all Web communication is a nontrivial task requiring a lot of careful thought, experimentation, and coordination. As such, crystal gazing for HTTP 2.0 timelines is dangerous business—it will be ready when it is ready. Having said that, the HTTP Working Group is making rapid progress. Its past and projected milestones are as follows:

- November 2009—SPDY protocol announced by Google.
- March 2012—call for proposals for HTTP 2.0.
- September 2012—first draft of HTTP 2.0.
- July 2013—first implementation draft of HTTP 2.0.
- April 2014—Working Group last call for HTTP 2.0.
- November 2014—submit HTTP 2.0 to IESG (Internet Engineering Steering Group) as a Proposed Standard.

SPDY was an experimental protocol developed at Google and announced in mid-2009, which later formed the basis of early HTTP 2.0 drafts. Many revisions and improvements later, as of late 2013, there is now an implementation draft of the protocol, and interoperability work is in full swing—recent Interop events featured client and server implementations from Microsoft Open Technologies, Mozilla, Google, Akamai, and other contributors. In short, all signs indicate the projected schedule is (for once) on track: 2014 should be the year for HTTP 2.0.

Making the Web (Even) Faster

With HTTP 2.0 deployed far and wide, can we kick back and declare victory? The Web will be fast, right? Well, as with any performance optimization, the moment one bottleneck is removed, the next one is unlocked. There is plenty of room for further optimization:

- HTTP 2.0 eliminates HOL blocking at the application layer, but it still exists at the transport (TCP) layer. Further, now that all of the streams can be multiplexed over a single connection, tuning congestion control, mitigating bufferbloat, and all other TCP optimizations become even more critical.
- TLS is a critical and largely unoptimized frontier: we need to reduce the number of handshake round-trips, upgrade outdated clients to get wider adoption, and improve client and server performance in general.
- HTTP 2.0 opens up a new world of research opportunities for optimal implementations of header-compression strategies, prioritization, and flow-control logic both on the client and server, as well as the use of server push.
- All existing Web applications will continue to work over HTTP 2.0—the servers will have to be upgraded, but otherwise the transport switch is transparent. That is not to say, however, that existing and new applications cannot be tuned to perform better over HTTP 2.0 by leveraging new functionality such as server push, prioritization, and so on. Web developers will have to develop new best practices, and revert and unlearn the numerous HTTP 1.1 workarounds they are using today.

In short, there is a lot more work to be done. HTTP 2.0 is a significant milestone that will help make the Web faster, but it is not the end of the journey.

Related articles on queue.acm.org

Improving Performance on the Internet
Tom Leighton
http://queue.acm.org/detail.cfm?id=1466449

High-Performance Websites
Steve Souders
http://queue.acm.org/detail.cfm?id=1466450

How Fast is Your Website?
Patrick Meenan
http://queue.acm.org/detail.cfm?id=2446236

References

1. Akamai. State of the Internet, 2013; http://www.akamai.com/stateoftheinternet/.
2. AT&T. Average speeds for AT&T LaptopConnect Devices, 2013; http://www.att.com/esupport/article.jsp?sid=64785&cv=820&_requestid=733221#fbid=vttq9CyA2iG and http://developer.att.com/home/develop/referencesandtutorials/whitepapers/BestPracticesFor3Gand4GAppDevelopment.pdf.
3. Belshe, M. More bandwidth doesn't matter (much), 2010; https://docs.google.com/a/chromium.org/viewerr?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2.
4. Grigorik, I. High-performance networking in Google Chrome, 2013; http://www.igvita.com/posa/high-performance-networking-in-google-chrome/.
5. HTTP Archive; http://www.httparchive.org/.
6. IETF HTTPbis Working Group. Charter, 2012; http://datatracker.ietf.org/doc/charter-ietf-httpbis/.
7. IETF HTTPbis Working Group. HTTP 2.0 specifications, 2013; http://tools.ietf.org/html/draft-ietf-httpbis-http2.
8. IETF HTTPbis Working Group. HPACK-Header Compression for HTTP/2.0, 2013; http://tools.ietf.org/html/draft-ietf-httpbis-header-compression.
9. IETF Network Working Group. Transport Layer Security (TLS) Application Layer Protocol Negotiation (ALPN) Extension, 2013; http://tools.ietf.org/html/draft-friedl-tls-applayerprotoneg.
10. Upson, L. Google I/O 2013 keynote address, 2010; http://www.youtube.com/watch?v=9pmPa_KxsAM.

Ilya Grigorik is a Web performance engineer and developer advocate at Google where he works to make the Web faster by building and driving adoption of performance best practices at Google and beyond.

DOI:10.1145/2534706.2534719

Article development led by acmqueue
queue.acm.org

Interfacing between languages is becoming more important.

BY DAVID CHISNALL

The Challenge of Cross-Language Interoperability

INTEROPERABILITY BETWEEN LANGUAGES has been a problem since the second programming language was invented. Solutions have ranged from language-independent object models such as Component Object Model (COM) and Common Object Request Broker Architecture (CORBA) to virtual machines (VMs) designed to integrate languages such as Java Virtual Machine (JVM) and Common Language Runtime (CLR).

With software becoming ever more complex and hardware less homogeneous, the likelihood of a single language being the correct tool for an entire program is lower than ever. As modern compilers become more modular, there is potential for a new generation of interesting solutions.

In 1961 the British company Stantec released a computer called the ZEBRA, which was interesting for a number of reasons, not least of which was its data flow-based instruction set. The ZEBRA was quite difficult to program with the full form of its native instruction set, so it also included a more conventional version, called Simple Code. This form came with some restrictions, including a limit of 150 instructions per program. The manual helpfully informs users that this is not a severe limitation, as it is impossible that someone would write a working program so complex that it would need more than 150 instructions.

Today, this claim seems ludicrous. Even simple functions in a relatively low-level language such as C have more than 150 instructions once they are compiled, and most programs are far more than a single function. The shift from writing assembly code to writing in a higher-level language dramatically increased the complexity of programs that were possible, as did various software engineering practices.

The trend toward increased complexity in software shows no sign of abating, and modern hardware creates new challenges. Programmers in the late 1990s had to target PCs at the low end that had an abstract model a lot like a fast PDP-11. At the high end, they would have encountered an abstract model like a very fast PDP-11, possibly with two to four (identical) processors. Now, mobile phones are starting to appear with eight cores with the same ISA (instruction set architecture) but different speeds, some other streaming processors optimized for different workloads (DSPs, GPUs), and other specialized cores.

The traditional division between high-level languages representing the class that is similar to a human's understanding of the problem domain and low-level languages representing the class similar to the hardware no longer applies. No low-level language has semantics that are close to a programmable data-flow processor, an x86 CPU,

IMAGE COLLAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

RedLine Smalltalk
Objective-C
Java Virtual Machine
Virtual Machines
CLR Common Language Runtime
Component Object Model
CORBA Common Object Request Broker Architecture

a massively multithreaded GPU, and a very long instruction word (VLIW) digital signal processor (DSP). Programmers wanting to get the last bit of performance out of the available hardware no longer have a single language they can use for all probable targets.

Similarly, at the other end of the abstraction spectrum, domain-specific languages are growing more prevalent. High-level languages typically trade generality for the ability to represent a subset of algorithms efficiently. More general-purpose high-level languages such as Java sacrifice the ability to manipulate pointers directly in exchange for providing the programmer with a more abstract memory model. Specialized languages such as SQL make certain categories of algorithms impossible to implement but make common tasks within their domain possible to express in a few lines.

You can no longer expect a nontrivial application to be written in a single language. High-level languages typically call code written in lower-level languages as part of their standard libraries (for example, GUI rendering), but adding calls can be difficult.

In particular, interfaces between two languages that are not C are often difficult to construct. Even relatively simple examples, such as bridging between C++ and Java, are not typically handled automatically and require a C interface. The Kaffe Native Interface[4] did provide a mechanism for doing this, but it was not widely adopted and had limitations.

The problem of interfacing between languages is going to become increasingly important to compiler writers over the coming years. It presents a number of challenges, detailed here.

The problem of interfacing between languages is going to become increasingly important to compiler writers over the coming years.

Object Model Differences

Object-oriented languages bind some notion of code and data together. Alan Kay, who helped develop object-oriented programming while at Xerox PARC, described objects as "simple computers that communicate via message passing." This definition leaves a lot of leeway for different languages to fill in the details:

- Should there be factory objects (classes) as first-class constructs in the language?
- If there are classes, are they also objects?
- Should there be zero (for example, Go), one (for example, Smalltalk, Java, JavaScript, Objective-C), or many (for example, C++, Self, Simula) superclasses or prototypes for an object?
- Is method lookup tied to the static type system (if there is one)?
- Is the data contained within an object of static or dynamic layout?
- Is it possible to modify method lookup at runtime?

The question of multiple inheritance is one of the most common areas of focus. Single inheritance is convenient, because it simplifies many aspects of the implementation. Objects can be extended just by appending fields; a cast to the supertype just involves ignoring the end, and a cast to a subtype just involves a check—the pointer values remain the same. Downcasting in C++ requires a complex search of the inheritance graph in the run-time type information via a runtime library function.

In isolation, both types of inheritance are possible to implement, but what happens if you want, for example, to expose a C++ object into Java? You could perhaps follow the .NET or Kaffe approach, and support direct interoperability with only a subset of C++ (Managed C++ or C++/CLI) that supports single inheritance only for classes that will be exposed on the Java side of the barrier.

This is a good solution in general: define a subset of one language that maps cleanly to another but can invoke the full power of the superset. This is the approach taken in Pragmatic Smalltalk:[5] Allow Objective-C++ objects (which can have C++ objects as instance variables and invoke their methods) to be exposed directly as if they were Smalltalk objects, sharing the same underlying representation.

This approach still provides a cognitive barrier, however. If you want to use a C++ framework directly, such as LLVM from Pragmatic Smalltalk or .NET, then you will need to write single-inheritance classes that encapsulate the multiple-inheritance classes the library uses for most of its core types.

Another possible approach would be to avoid exposing any fields within the objects and just expose each C++

class as an interface. This would, however, make it impossible to inherit from the bridged classes without special compiler support to understand that some interfaces came along with implementation.

Although complex, this is a simpler system than interfacing between languages that differ on what method lookup means. For example, Java and Smalltalk have almost identical object and memory models, but Java ties the notion of method dispatch to the class hierarchy, whereas in Smalltalk two objects can be used interchangeably if they implement methods with the same names.

This is a problem encountered by RedLine Smalltalk,[1] which compiles Smalltalk to run on JVM. Its mechanism for implementing Smalltalk method dispatch involves generating a Java interface for each method and then performing a cast of the receiver to the relevant interface type before dispatch. Sending messages to Java classes requires extra information, because existing Java classes do not implement this; thus, RedLine Smalltalk must fall back to using Java's Reflection APIs.

The method lookup for Smalltalk (and Objective-C) is more complex, because there are a number of second-chance dispatch mechanisms that are either missing or limited in other languages. When compiling Objective-C to JavaScript, rather than using the JavaScript method invocation, you must wrap each Objective-C message send in a small function that first checks if the method actually exists and, if it does not, calls some lookup code.

This is relatively simple in JavaScript because it handles variadic functions in a convenient way: if a function or method is called with more arguments than it expects, then it receives the remainder as an array that it can expect. Go does something similar. C-like languages just put them on the stack and expect the programmer to do the right thing with no error checking.

Memory Models

The obvious dichotomy in memory models is between automatic and manual deallocation. A slightly more important concern is the difference between deterministic and nondeterministic destruction.

It is possible to run C with the Boehm-Demers-Weiser garbage collector[3] without problems in many cases (unless you run out of memory and have a lot of integers that look like pointers). It is much harder to do the same for C++, because object deallocation is an observable event. Consider the code shown in Figure 1.

The `LockHolder` class defines a very simple object; a mutex passes into the object, which then locks the mutex in its constructor and unlocks it in the destructor. Now, imagine running this same code in a fully garbage-collected environment—the time at which the destructor runs is not defined.

This example is relatively simple to get right. A garbage-collected C++ implementation is required to run the destructor at this point but not to deallocate the object. This idiom is not available in languages designed to support garbage collection from the start. The fundamental problem with mixing them is not determining who is responsible for releasing memory; rather, it is that code written for one model expects deterministic operation, whereas code written for the other does not.

There are two trivial approaches to implementing garbage collection for C++: the first is to make the `delete` operator invoke destructors but not reclaim the underlying storage; the other is to make `delete` a no-op and call destructors when the object is detected as unreachable.

Destructors that call only `delete` are the same in both cases: they are effectively no-ops. Destructors that release other resources are different. In the first case, they run deterministically but will fail to run if the programmer does not explicitly delete the relevant object. In the second case, they are guaranteed to run eventually but not necessarily by the time the underlying resource is exhausted.

Additionally, a fairly common idiom in many languages is a self-owned object that waits for some event or performs a long-running task and then fires a callback. The receiver of the callback is then responsible for cleaning up the notifier. While it is live, it is disconnected from the rest of the object graph and so appears to be garbage. The collector must be explicitly told that it is not. This is the opposite of the pattern in languages without automatic garbage collection, where objects are assumed to be live unless the system is told otherwise. (Hans Boehm discussed some of these issues in more detail in a 1996 paper.[2])

All of these problems were present with Apple's ill-fated (and, thankfully, no longer supported) attempt to add garbage collection to Objective-C. A lot of Objective-C code relies on running code in the `-dealloc` method. Another issue was closely related to the problem of interoperability. The implementation supported both traced and untraced memory but did not expose this information in the type system. Consider the snipped noted in Figure 2.

Figure 1. C++ code using deterministic automatic deallocation to relinguish a lock.

```
{
        LockHolder l( mutex );
        /* Do stuff that requires mutex to be locked */
}
```

Figure 2. Objective-C code demonstrating the problems with retrofitting garbage collection to an existing language.

```
void allocateSomeObjects (id * buffer , int count )
{
        for (int i=0 ; i< count ; i++)
        {
                buffer [i] = [ SomeClass new ];
        }
}
```

In garbage-collected mode, it is impossible to tell if this code is correct. Whether it is correct or not depends on the caller. If the caller passes a buffer allocated with `NSAllocateCollectable()`, with `NSScannedOption` as the second parameter, or with a buffer allocated on the stack or in a global in a compilation unit compiled with garbage-collection support, then the objects will last (at least) as long as the buffer. If the caller passes a buffer that was allocated with `malloc()` or as a global in a C or C++ compilation unit, then the objects will (potentially) be deallocated before the buffer. The *potentially* in this sentence makes this a bigger problem: because it is nondeterministic, it is hard to debug.

The Automatic Reference Counting (ARC) extensions to Objective-C do not provide complete garbage collection (they still allow garbage cycles to leak), but they do extend the type system to define the ownership type for such buffers. Copying object pointers to C requires the insertion of an explicit cast containing an ownership transfer.

Reference counting also solves the determinism problem for acyclic data. In addition, it provides an interesting way of interoperating with manual memory management: by making `free()` decrement the reference count. Cyclic (or potentially cyclic) data structures require the addition of a cycle detector. David F. Bacon's team at IBM has produced a number of designs for cycle detectors[8] that allow reference counting to be a full garbage-collection mechanism, as long as pointers can be accurately identified.

Unfortunately, cycle detection involves walking the entire object graph from a potentially cyclic object. Some simple steps can be taken to lessen this cost. The obvious one is to defer it. An object is only potentially part of a cycle if its reference count is decremented but not deallocated. If it is later incremented, then it is not part of a garbage cycle (it may still be part of a cycle, but you do not care yet). If it is later deallocated, then it is acyclic.

The longer you defer cycle detection, the more nondeterminism you get, but the less work the cycle detector has to do.

Exceptions and Unwinding

These days, most people think of exceptions in the sense popularized by C++: something that is roughly equivalent to `setjmp()` and `longjmp()` in C, although possibly with a different mechanism.

A number of other mechanisms for exceptions have been proposed. In Smalltalk-80, exceptions are implemented entirely in the library. The only primitive the language provides is that when you explicitly return from a closure, you return from the scope in which the closure was declared. If you pass a closure down the stack, then a return will implicitly unwind the stack.

When a Smalltalk exception occurs, it invokes a handler block on the top of the stack. This may then return, forcing the stack to unwind, or it may do some cleanup. The stack itself is a list of activation records (which are objects) and therefore may do something more complex. Common Lisp provides a rich set of exceptions too, including those that support resuming or restarting immediately afterward.

Exception interoperability is difficult even within languages with similar exception models. For example, C++ and Objective-C both have similar notions of an exception, but what should a C++ catch block that expects to catch a `void*` do when it encounters an Objective-C object pointer? In the GNUstep Objective-C runtime[6], we chose not to catch it after deciding not to emulate Apple's behavior of a segmentation fault. Recent versions of OS X have adopted this behavior, but the decision is somewhat arbitrary.

Even if you do catch the object pointer from C++, that does not mean you can do anything with it. By the time it is caught, you have lost all of the type information and have no way of determining if it is an Objective-C object.

Subtler issues creep in when you start to think about performance. Early versions of VMKit[7] (which implements Java and CLR VMs on top of LLVM) used the zero-cost exception model designed for C++. This is *zero cost* because entering a try block costs nothing. When throwing an exception, however, you must parse some tables that describe how to unwind the stack, then call into a personality function for each stack frame to decide whether (and where) the exception should be caught.

This mechanism works very well for C++, where exceptions are rare, but Java uses exceptions to report lots of fairly common error conditions. In benchmarks, the performance of the unwinder was a limiting factor. To avoid this, the calling convention was modified for methods that were likely to throw an exception. These functions returned the exception as a second return value (typically in a different register), and every call just had to check that this register contained 0 or jump to the exception handling block if it did not.

This is fine when you control the code generator for every caller, but this is not the case in a cross-language scenario. You might address the issue by adding another calling convention to C that mirrors this behavior or that provides something like the multiple-return-values mechanism commonly used in Go for returning error conditions, but that would require every C caller to be aware of the foreign language semantics.

Mutability and Side Effects

When you start to include functional languages in the set with which you wish to interoperate, the notion of mutability becomes important. A language such as Haskell has no mutable types. Modifying a data structure in place is something the compiler may do as an optimization, but it is not something exposed in the language.

This is a problem encountered by F#, which is sold as a dialect of OCaml and can integrate with other .NET languages, use classes written in C#, and so on. C# already has a notion of mutable and immutable types. This is a very powerful abstraction, but an immutable class is simply one that does not expose any fields that are not read only, and a read-only field may contain references to objects that (via an arbitrary chain of references) refer to mutable objects whose state may be changed out from under the functional code. In other languages, such as C++ or Objective-C, mutability is typically implemented within the class system by defining some classes that are immutable, but there is no language support and no easy way of determining whether an object is mutable.

C and C++ have a very different concept of mutability in the type system provided by the language: a particular reference to an object may or may not modify it, but this does not mean the object itself will not change. This, combined with the deep copying problem, makes interfacing functional and object-oriented languages a difficult problem.

Monads provide some tantalizing possibilities for the interface. A monad is an ordered sequence of computational steps. In an object-oriented world, this is a series of message sends or method invocations. Methods that have the C++ notion of const (that is, do not modify the state of the object) may be invoked outside of the monad, and so are amenable to speculative execution and backtracking, whereas other methods should be invoked in a strict sequence defined by the monad.

Models of Parallelism

Mutability and parallelism are closely related. The cardinal rule for writing maintainable, scalable, parallel code is that no object may be both mutable and aliased. This is trivial to enforce in a purely functional language: no object is mutable at all. Erlang makes one concession to mutability, in the form of a process dictionary—a mutable dictionary that is accessible from only the current Erlang process and so can never be shared.

Interfacing languages with different notions of what can be shared presents some unique problems. This is interesting for languages intended to target massively parallel systems or GPUs, where the model for the language is intimately tied to the underlying hardware.

This is the issue encountered when trying to extract portions of C/C++/Fortran programs to turn into OpenCL. The source languages typically have in-place modification as the fastest way of implementing an algorithm, whereas OpenCL encourages a model where a source buffer is processed to generate an output buffer. This is important because each kernel runs in parallel on many inputs; thus, for maximum throughput they should be independent.

How to expose multiple processing units with very different abstract machine models to the programmer is an interesting research problem. It is very difficult to provide a single language that efficiently captures the semantics.

In C++, however, ensuring that two pointers do not alias is nontrivial. The restrict keyword exists to allow programmers to provide this annotation, but it is impossible in the general case for a compiler to check that it is correctly used.

Efficient interoperability is very important for heterogeneous multicore systems. On a traditional single-core or SMP (symmetric multiprocessing) computer, there is a one-dimensional spectrum between high-level languages that are close to the problem domain and low-level languages that are close to the architecture. On a heterogeneous system, no one language is close to the underlying architecture, as the difficulty of running arbitrary C/C++ and Fortran code on GPUs has shown.

Current interfaces—for example, OpenCL—are a long way from ideal. The programmer must write C code to manage the creation of a device context and the movement of data to and from the device, and then write the kernel in OpenCL C. The ability to express the part that runs on the device in another language is useful, but when most of the code for simple operations is related to the boundary between the two processing elements rather than the work done on either side, then something is wrong.

How to expose multiple processing units with very different abstract machine models to the programmer is an interesting research problem. It is very difficult to provide a single language that efficiently captures the semantics. Thus, this problem becomes one of interoperability between specialized languages. This is an interesting shift in that domain-specific languages, which are traditionally at the high-level end of the spectrum, now have an increasing role to play as low-level languages.

The VM Delusion

The virtual machine is often touted as a way of addressing the language interoperability problem. When Java was introduced, one of the promises was that you would soon be able to compile all of your legacy C or C++ code and run it in JVM alongside Java, providing a clean migration path. Today, Ohloh.net (which tracks the number of lines of code available in public open source repositories) reports four billion lines of C code, and around 1.5 billion each of C++ and Java. While other languages such as Scala (almost six million lines of code tracked by Ohloh.net) run in JVM, legacy low-level languages do not.

Worse, calling native code from Java is so cumbersome (both in terms of cognitive and runtime overhead) that developers end up writing applications in C++ rather than face calling into a C++ library from Java. Microsoft's CLR did a little better, allowing code written in a subset of C++ to run; it makes calling out to native libraries easier but still provides a wall.

This approach has been a disaster for languages such as Smalltalk that do not have large companies backing them. The Smalltalk VM provides some advantages that neither CLR nor JVM provides in the form of a persistence model and reflective development environment, but it also forms a very large PLIB (programming language interoperability barrier) by dividing the world into things that are inside and things that are outside the box.

This gets even more complex once you have two or more VMs and now have the problem of source-language interoperability and the (very similar) problem of interoperability between the two VMs, which are typically very low-level programming languages.

The Path Forward

Many years ago the big interoperability question of the day was C and Pascal—two languages with an almost identical abstract machine model. The problem was that Pascal compilers pushed their parameters onto the stack left to right (because that required fewer temporaries), whereas C compilers pushed them right to left (to ensure the first ones were at the top of the stack for variadic functions).

This interoperability problem was largely solved by the simple expedient of defining calling conventions as part of the platform application binary interface (ABI). No virtual machine or intermediate target was required, nor was any source-to-source translation. The equivalent of the virtual machine is defined by the ABI and the target machine's ISA.

Objective-C provides another useful case study. Methods in Objective-C use the C calling convention, with two hidden parameters (the object and the selector, which is an abstract form of the method name) passed first. All parts of the language that do not trivially map to the target ABI or ISA are factored out into library calls. A method invocation is implemented as a call to the `objc_msgSend()` function, which is implemented as a short assembly routine. All of the introspection works via the mechanism of calls to the runtime library.

We have used GNUstep's Objective-C runtime to implement front ends for dialects of Smalltalk and JavaScript in LanguageKit. This uses LLVM, but only because having a low-level intermediate representation permits optimizations to be reused between compilers: the interoperability happens in the native code. This runtime also supports the blocks ABI defined by Apple; therefore, closures can be passed between Smalltalk and C code.

Boehm garbage collector (GC) and Apple AutoZone both aimed to provide garbage collection in a library form, with different requirements. Can concurrent compacting collectors be exposed as libraries, with objects individually marked as nonmovable when they are passed out to low-level code? Is it possible to enforce mutability and concurrency guarantees in an ABI or library? These are open problems, and the availability of mature libraries for compiler design makes them interesting research questions.

Perhaps more interesting is the question of how many of these can be sunk down into the hardware. In Crash-worthy Trustworthy Systems R&D (CTSRD), a joint project between SRI International and the University of Cambridge Computer Laboratory, researchers have been experimenting with expressing fine-grained memory protection into the hardware, which they hope will provide more efficient ways of expressing certain language memory models. This is a start, but there is a lot more potential for providing richer feature sets for high-level languages in silicon, something that was avoided in the 1980s because transistors were scarce and expensive resources. Now transistors are plentiful but power is scarce, so the trade-offs in CPU design are very different.

The industry has spent the past 30 years building CPUs optimized for running languages such as C, because people who needed fast code used C (because people who designed processors optimized them for C, because...). Maybe the time has come to start exploring better built-in support for common operations in other languages. The RISC project was born from looking at the instructions that a primitive compiler generated from compiling C code. What would we end up with if we started by looking at what a naïve JavaScript or Haskell compiler would emit?

Related articles on queue.acm.org

The Robustness Principle Reconsidered
Eric Allman
http://queue.acm.org/detail.cfm?id=1999945

Erlang for Concurrent Programming
Jim Larson
http://queue.acm.org/detail.cfm?id=1454463

The Rise and Fall of CORBA
Michi Henning
http://queue.acm.org/detail.cfm?id=1142044

References

1. Allen, S. RedLine Smalltalk. Presented at the International Smalltalk Conference (2011).
2. Boehm, H.-J. Simple garbage-collector-safety. *ACM SIGPLAN Notices 31*, 5 (1996), 89–98.
3. Boehm, H.-J. and Weiser, M. Garbage collection in an uncooperative environment. *Software Practice and Experience 18*, 9 (1988), 807–820.
4. Bothner, P. and Tromey, T. Java/C++ integration (2001); http://per.bothner.com/papers/UsenixJVM01/CNI01.pdf/ and http://gcc.gnu.org/java/papers/native++.html/
5. Chisnall, D. 2012. Smalltalk in a C world. In *Proceedings of the International Workshop on Smalltalk Technologies* (2012), 4:1–4:12.
6. Chisnall, D. A New objective-C runtime: From research to production. *ACM Queue*, (2012); http://queue.acm.org/detail.cfm?id=2331170/
7. Geoffray, N., Thomas, G., Lawall, J., Muller, G. and Folliot, B. VMKit: A substrate for managed runtime environments. *ACM SIGPLAN Notices 45*, 7 (2010), 51–62.
8. Paz, H., Petrank, E., Bacon, D. F., Kolodner, E.K., Rajan, V.T. An efficient on-the-fly cycle collection. In *Proceedings of the 14th International Conference on Compiler Construction*. Springer-Verlag, Berlin, Heidelberg, 2001, 156–171.

Portions of this work were sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of DARPA or the U.S. Department of Defense.

David Chisnall is a researcher at the University of Cambridge, where he works on programming language design and implementation. He spent several years consulting, during which time he also wrote books on Xen and the Objective-C and Go programming languages. He also contributes to the LLVM, Clang, FreeBSD, GNUstep, and Étoilé open source projects.

DOI:10.1145/2534706.2534720

Article development led by acmqueue
queue.acm.org

The increasing significance of intermediate representations in compilers.

BY FRED CHOW

Intermediate Representation

PROGRAM COMPILATION IS a complicated process. A compiler is a software program that translates a high-level source-language program into a form ready to execute on a computer. Early on in the evolution of compilers, designers introduced intermediate representations (IRs, also commonly called intermediate languages) to manage the complexity of the compilation process. The use of an IR as the compiler's internal representation of the program enables the compiler to be broken up into multiple phases and components, thus benefiting from modularity.

An IR is any data structure that can represent the program without loss of information so that its execution can be conducted accurately. It serves as the common interface among the compiler components. Since its use is internal to a compiler, each compiler is free to define the form and details of its IR, and its specification needs to be known only to the compiler writers. Its existence can be transient during the compilation process, or it can be output and handled as text or binary files. An IR should be general so that it is capable of representing programs translated from multiple languages. Compiler writers traditionally refer to the semantic content of programming languages as being high. The semantic content of machine-executable code is considered low because it has retained only enough information from the original program to allow its correct execution. It would be difficult (if not impossible) to recreate the source program from its lower form. The compilation process entails the gradual lowering of the program representation from high-level human programming constructs to low-level real or virtual machine instructions (see Figure 1). In order for an IR to be capable of representing multiple languages, it needs to be closer to the machine level to represent the execution behavior of all the languages. A longer code sequence usually accompanies machine-executable code because it reflects the details of the machines on which execution takes place.

A well-designed IR should be translatable into different forms for execution on multiple platforms. For execution on a target processor or CPU, it needs to be translated into the assembly language of that processor, which usually is a one-to-one mapping to the processor's machine instructions. Since there are different processors with different instruction set architectures (ISAs), the IR needs to be at a higher level than typical machine instructions where it does not assume any special machine characteristic.

Using an IR enables a compiler to support multiple frontends that translate from different programming languages and multiple backends to generate code for different processor targets (see Figure 2). The execution platform can also be interpretive in the sense that its execution is conducted by a software program or virtual machine. In such cases, the medium of execution can be at a level higher than assembly code, while being lower or at the same level as the IR.

The adoption of IRs enables the modularization of the compilation process into the frontend, the middle end, and the backend. The frontend specializes in handling the programming language aspects of the compiler. A programming language implementer only needs to realize the accurate translation of the language to an IR before declaring the work complete.

The backend takes into account the particulars of the target machine and translates the IR into the machine instructions to be executed on the hardware. It also transforms the code to take advantage of any hardware features that benefit performance. By starting its translation from the IR, the backend in effect supports the different languages that produce the IR.

The middle end is where target-independent optimizations take place. The middle-end phases perform different transformations on the IR so the program can run more efficiently. Because optimizations on the IR usually benefit all targets, the IR has assumed the significant role of being the medium for performing target-independent optimizing transformations. In this role, its design and specification become even more important. It needs to encode any source-program information that is helpful to the optimization tasks. The IR's design has a bearing on optimization efficiency, and optimization is the most time-consuming part of the compilation process. In modern-day compilers, the IR dictates the infrastructure and overall engineering of the compiler. Any major change to the IR could imply a substantial overhaul in the compiler implementation.

The Different IR Forms

The minimal requirement of an IR is to provide enough information for the correct execution of the original program. Each instruction in an IR typically represents one simple operation. An IR should have fewer kinds of constructs compared with any typical programming language, because it does not need to be feature rich to facilitate programming use by humans. Compilers like to see the same pro-

Figure 1. The different levels of program representation.

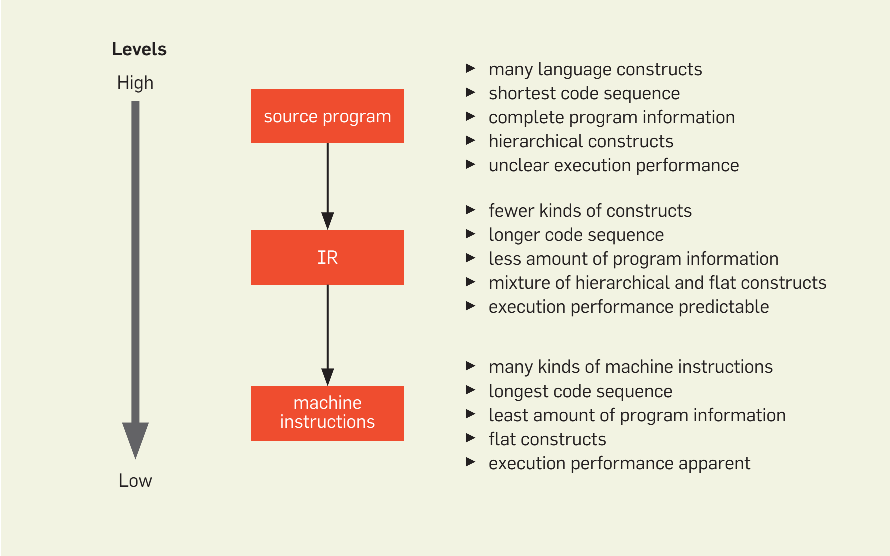
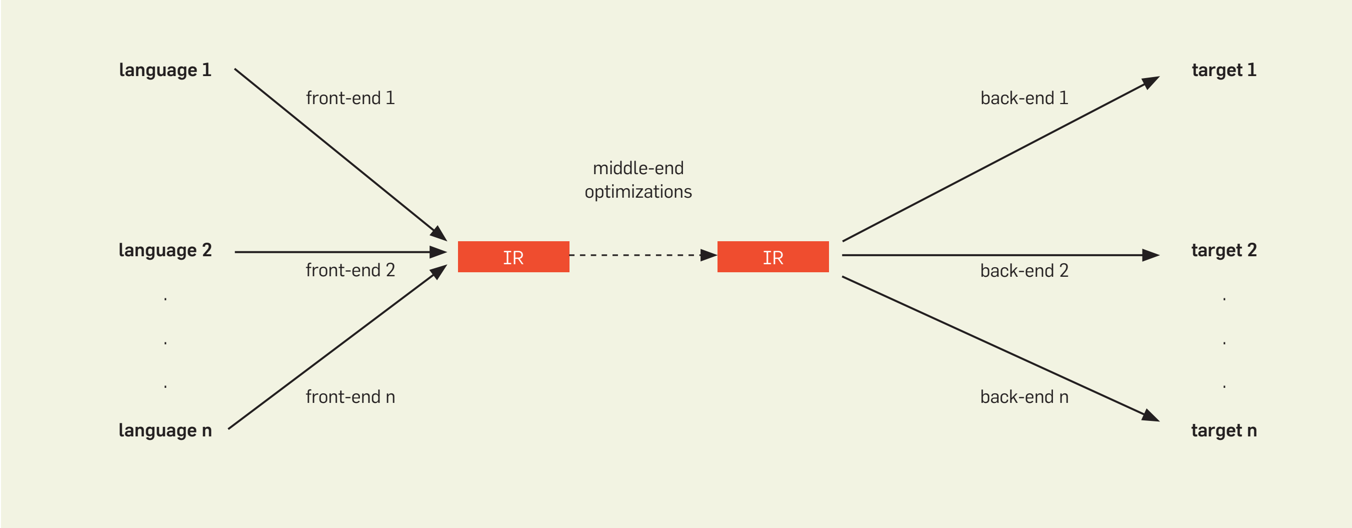

Levels
High
Low
source program
IR
machine instructions
many language constructs
shortest code sequence
complete program information
hierarchical constructs
unclear execution performance
fewer kinds of constructs
longer code sequence
less amount of program information
mixture of hierarchical and flat constructs
execution performance predictable
many kinds of machine instructions
longest code sequence
least amount of program information
flat constructs
execution performance apparent

Figure 2. A compiler system supporting multiple languages and multiple targets.



gramming constructs or idioms being translated to uniform code sequences in the IR, regardless of their source languages, programming styles, or the ways the programmers choose to code them. Imposing canonical forms in IRs reduces the variety of code patterns that the compiler has to deal with in performing code generation and optimization. Because of its finer-grained representation of the program, an IR's instructions may map many-to-one to a machine instruction because one machine instruction may perform multiple operations, as in multiply-add or indexed addressing.

The form of an IR can be classified as either hierarchical or flat. A hierarchical IR allows nested structures. In a typical programming language, both program control flows (for example, if-then-else, do-loops) and arithmetic expressions are nested structures. A hierarchical IR is thus closer in form to the typical programming language, and is regarded as being at a higher level. A hierarchical IR can be represented internally in the form of trees (the data structure preferred by compilers) without loss of accuracy.

A flat IR is often viewed as the instructions of an abstract or virtual machine. The instructions are executed sequentially, as in a typical processor, and control flows are specified by branch or jump instructions. Each instruction takes a number of operands and produces a result. Such IRs are often specified as compilation targets in the teaching of compiler construction.

Lying somewhere between hierarchical and flat IRs is the language of an abstract stack machine. In a stack machine, each operand for arithmetic computation is specified by an instruction that pushes the operand onto the stack. Each arithmetic expression evaluation is done on the operands that are popped off the top of the stack, and the subsequent result is pushed back on the stack. The form of the IR is flat, with control flow represented by explicit branch instructions, but the instruction sequence for arithmetic computation can be regarded as corresponding to the reverse Polish notation, which can be easily represented internally in a tree data structure. Using the language of a stack machine as the IR has been a common practice dating from the first IR defined for the Pascal language, called p-code,[1] to the current-day Java bytecode[6] or Common Intermediate Language (CIL[2]).

An IR should have fewer kinds of constructs compared with any typical programming language, because it does not need to be feature rich to facilitate programming use by humans.

There is information complementary to the IR that serves purposes other than representing code execution. The compiler compiles the namespace in the original program into a collection of symbol names. Variables, functions, and type information belong to these symbol tables, and they can encode information that governs the legality of certain optimizing transformations. They also provide information needed by various tools such as debuggers and program analyzers. The symbol tables can be considered adjunct to the IRs.

C has been used as the translation target of many programming languages because of its widespread use as a system programming language and its ability to represent any machine operation. C can be regarded as an IR because of its lower level relative to most languages, but it was not designed for easy manipulation by compilers or to be directly interpreted. In spite of this, many IRs have been designed by closely modeling the C language semantics. In fact, a good IR can be constructed by carefully stripping away C's high-level control-flow constructs and structured data types, leaving behind only its primitives. Many IRs can also be translated to C-like output for easy perusal by compiler developers. Such C-like IRs, however, usually cannot be translated to C programs that can be recompiled because of C's deficiencies in representing certain programming concepts such as exception handling, overflow checking, or multiple entry points to a function.

IRs for Program Delivery

With the widespread use of networked computers, people soon understood the advantage of an execution medium that is processor- and operating-system-neutral. The distribution and delivery process is easier with programs that can run on any machine. This write-once, run-anywhere approach can be realized with the virtual machine execution model to accommodate the diversity of system hardware.

Interpretive execution contributes to some loss of performance compared with compiled execution, and initially it made sense only for applications that are not compute intensive. As machines become faster and faster, however, the advantages of the write-once, run-anywhere approach outweigh potential performance loss in many applications. This gave rise to the popularity of languages such as Java that can be universally deployed. The Java language defines the Java bytecode as its distribution medium, which is a form of IR. Java bytecode can be run on any platform as long as the Java virtual machine (JVM) software is installed. Another example is CIL, which is the IR of the Common Language Infrastructure (CLI) runtime environment used by the .NET Framework.

With the growth of the mobile Internet, applications are often downloaded to handheld devices to be run instantly. Since IRs take up less storage than machine executables, they result in lower network transmission overhead, as well as enabling hardware-independent program distribution.

Just-In-Time Compilation

As the virtual machine execution model gained widespread acceptance, it became important to find ways of speeding up the execution. One method is just-in-time (JIT) compilation, also known as dynamic compilation, which improves the performance of interpreted programs by compiling them during execution into native code to speed up execution on the underlying machine. Since compilation at runtime incurs overhead that slows down the program execution, it would be prudent to take the JIT route only if there is a high likelihood that the resultant reduction in execution time more than offsets the additionally incurred compilation time. In addition, the dynamic compiler cannot spend too much time optimizing the code, as optimization incurs much greater overhead than translation to native code. To restrain the overhead caused by dynamic compilation, most JIT compilers compile only the code paths that are most frequently taken during execution.

Computer manufacturers have come to the realization that further increases in computing performance can no longer rely on increases in clock frequency.

Dynamic compilation does have a few advantages over static compilation. First, dynamic compilation can use real-time profiling data to optimize the generated code more effectively. Second, if the program behavior changes during execution, the dynamic compiler can recompile to adjust the code to the new profile. Finally, with the prevalent use of shared (or dynamic) libraries, dynamic compilation has become the only safe means of performing whole program analysis and optimization, in which the scope of compilation spans both user and library code. JIT compilation has become an indispensable component of the execution engine of many virtual machines that take IRs as input. The goal is to make the performance of programs built for machine-independent distribution approach that of native code generated by static compilers.

In recent years, computer manufacturers have come to the realization that further increases in computing performance can no longer rely on increases in clock frequency. This has given rise to special-purpose processors and coprocessors, which can be digital signal processors (DSPs), GPUs, or accelerators implemented in ASIC or field-programmable gate array (FPGA). The computing platform can even be heterogeneous where different types of computation are handed off to different types of processors, each having different instruction sets. Special languages or language extensions such as CUDA,[3] OpenCL,[8] and Hybrid Multicore Parallel Programming (HMPP),[4] with their underlying compilers, have been designed to make it easier for programmers to derive maximum performance in a heterogeneous setting.

Because these special processors are designed to increase performance, programs must be compiled to execute in their native instructions. As the proliferation of special-purpose hardware gathered speed, it became impossible for a compiler supplier to provide customized support for the variety of processors that exist in the market or are about to emerge. In this setting, the custom hardware manufacturer is responsible for providing the backend compiler that compiles the IR to the custom machine instructions, and platform-independent program delivery has become all the more important. In practice, the compilation can be effected earlier at installation time or at program loading instead of during execution. Nowadays, the term *AOT* (ahead-of-time), in contrast with JIT, characterizes the compilation of IRs into machine code before its execution. Whether it is JIT or AOT, however, IRs obviously play an enabling role in this new approach to providing high-performance computing platforms.

Standardizing IRs

So far, IRs have been linked to individual compiler implementations because most compilers are distinguished by the IRs they use. IRs are translatable, however, and it is possible to translate the IR of compiler A to that of compiler B, so compiler B can benefit from the work in compiler A. With the trend toward open source software in the past two decades, more and more compilers have been open sourced.[9] When a compiler becomes open source, it exposes its IR definition to the world. As the compiler's developer community grows, it has the effect of promoting its IR. Using an IR, however, is subject to the terms of its compiler's open source license, which often prohibits mixing it with other types of open source licenses. In case of licensing conflicts, special agreements need to be worked out with the license providers before such IR translations can be realized. In sum, IR translation enables collaboration between compilers.

Java bytecode is the first example of an IR with an open standard definition that is independent of compilers, because JVM is so widely accepted that it has spawned numerous compiler and VM implementations. The prevalence of JVM has led to many other languages being translated to Java bytecode,[7] but because it was originally defined to serve only the Java language, support for high-level abstractions not present in Java are either not straightforward or absent. This lack of generality limits the use of Java bytecode as a universal IR.

Because IRs can solve the object-code compatibility issue among different processors by simplifying program delivery while enabling maximum compiled-code performance on each processor, standardizing on an IR would serve the computing industry well. Experience tells us that it takes time for all involved parties to agree on a standard; most existing standards have taken years to develop, and sometimes, competing standards take time to consolidate into one. The time is ripe to start developing an IR standard. Once such a standard is in place, it will not stifle innovation as long as it is being continuously extended to capture the latest technological trends.

A standard IR will solve two different issues that have persisted in the computing industry:

▸ *Software compatibility.* Two pieces of software are not compatible when they are in different native code of different ISAs. Even if their ISAs are the same, they can still be incompatible if they have been built using different application binary interfaces (ABIs) or under different operating systems with different object file formats. As a result, many different incompatible software ecosystems exist today. The computing industry would be well served by defining a standard software distribution medium that is acceptable by most if not all computing platforms. Such a distribution medium can be based on the IR of an abstract machine. It will be rendered executable on a particular platform through AOT or JIT compilation. A set of compliance tests can be specified. Software vendors will need to distribute their software products only in this medium. Computing devices supporting this standard will be able to run all software distributed in this form. This standardized software ecosystem will create a level playing field for manufacturers of different types of processors, thus encouraging innovation in hardware.

▸ *Compiler interoperability.* The field of compilation with optimization is a conundrum. No single compiler can claim to excel in everything. The algorithm that a compiler uses may work well for one program but not so well for another. Thus, developing a compiler requires a huge effort. Even for a finished compiler, there may still be endless enhancements deemed desirable. Until now, each production-quality compiler has been operating on its own. This article has discussed IR translation as a way of allowing compilers to work together. A standard IR, if adopted by compilers, will make it possible to combine the strengths of the different compilers that use it. These compilers no longer need to incorporate the full compilation functionalities. They can be developed and deployed as compilation modules, and their creators can choose to make the modules either proprietary or open source. If a compiler module desires to operate using its own unique internal program representation, it can choose to use the standard IR only as an interchange format. A standard IR lowers the entry barrier for compiler writers, because their projects can be conceived at smaller scales, allowing each compiler writer to focus on his or her specialties. An IR standard also makes it easier to do comparisons among the compilers when they produce the same IR as output, which will lead to more fine-tuning. An IR standard will have the potential to revolutionize today's compiler industry and will serve the interests of compiler writers very well.

Two visions for an IR standard are outlined here: the first is centered on the computing industry, the second on the compiler industry. The first emphasizes the virtual machine aspect, and the second focuses on providing good support to the different aspects of compilation. Because execution requires less program information than compilation, the second goal will require greater content in the IR definition compared with the first goal. In other words, an IR standard that addresses the first goal may not fulfill the needs of the second. It is also hard to say at this point if one well-defined IR standard can fulfill both purposes at the same time.

The Heterogeneous System Architecture (HSA) Foundation was formed in 2012 with the charter to make programming heterogeneous devices dramatically easier by putting forth royalty-free specifications and open source software.[5] Its members intend to build a heterogeneous software ecosystem rooted in open royalty-free industry standards.

Recently, the foundation put forth a specification for HSAIL (HSA Intermediate Language), which is positioned as the ISA of a HSAIL virtual machine for any computing device that plans to adhere to the standard. HSAIL is quite low level, somewhat analogous to the assembly language of a RISC machine. It also assumes a specific program and memory model catering to heterogeneous platforms where multiple ISAs exist, with one specified as the host. It also specifies a model of parallel processing as part of the virtual machine.

Although HSAIL is aligned with the vision of enabling a software ecosystem based on a virtual machine, its requirements are too strong and lack generality, and thus will limit its applicability to the specific segment of the computing industry that it targets. Though HSAIL serves as the compilation target for compiler developers, it is unlikely that any compiler will adopt HSAIL as an IR during compilation because of the lack of simplicity in the HSAIL virtual machine. It is a step in the right direction, however.

IR Design Attributes

In conclusion, here is a summary of the important design attributes of IRs and how they pertain to the two visions discussed here. The first five attributes are shared by both visions.

▸ *Completeness.* The IR must provide clean representation of all programming language constructs, concepts, and abstractions for accurate execution on computing devices. A good test of this attribute is whether it is easily translatable both to and from popular IRs in use today for various programming languages.

▸ *Semantic gap.* The semantic gap between the source languages and the IR must be large enough so that it is not possible to recover the original source program, for the sake of intellectual property rights protection. This implies the level of the IR must be low.

▸ *Hardware neutrality.* The IR must not have built-in assumption of any special hardware characteristic. Any execution model apparent in the IR should be a reflection of the programming language and not the hardware platform. This ensures it can be compiled to the widest range of machines, and implies the level of the IR cannot be too low.

▸ *Manually programmable.* Programming in IRs is similar to assembly programming. This gives programmers the choice to hand-optimize their code. It is also a convenient feature that helps compiler writers during compiler development. A higher-level IR is usually easier to program.

▸ *Extensibility.* As programming languages continue to evolve, there will be demands to support new programming paradigms. The IR definition should provide room for extensions without breaking compatibility with earlier versions.

From the compiler's perspective, there are three more attributes that are important considerations for the IR to be used as program representation during compilation:

▸ *Simplicity.* The IR should have as few constructs as possible while remaining capable of representing all computations translated from programming languages. Compilers often perform a process called canonicalization that massages the input program into canonical forms before performing various optimizations. Having the fewest possible ways of representing a computation is actually good for the compiler, because there are fewer code variations for the compiler to cover.

▸ *Program information.* The most complete program information exists at the source form in which the program was originally written, some of which is derived from programming language rules. Translation out of the programming language will contribute to information loss, unless the IR provides ways of encoding the escaped information. Examples are high-level types and pointer aliasing information, which are not needed for program execution but affect whether certain transformations can be safely performed during optimization. A good IR should preserve any information in the source program that is helpful to compiler optimization.

▸ *Analysis information.* Apart from information readily available at the program level, program transformations and optimizations rely on additional information generated by the compiler's analysis of the program. Examples are data dependency, use-def and alias analysis information. Encoding such information in the IR allows it to be usable by other compiler components, but such information can also be invalidated by program transformations. If the IR encodes such analysis information, it needs to be maintained throughout the compilation, which puts additional burdens on the transformation phases. Thus, whether or not to encode information that can be gathered via program analysis is a judgment call. For the sake of simplicity, it can be left out or made optional.

A standard for a universal IR that enables target-independent program binary distribution and is usable internally by all compilers may sound idealistic, but it is a good cause that holds promises for the entire computing industry. C

Related articles on queue.acm.org

All Your Database Are Belong to Us
Erik Meijer
http://queue.acm.org/detail.cfm?id=2338507

Stream Processors: Progammability and Efficiency
William J. Dally, Ujval J. Kapasi, Brucek Khailany, Jung Ho Ahn, and Abhishek Das
http://queue.acm.org/detail.cfm?id=984486

Software Development with Code Maps
Robert DeLine, Gina Venolia, and Kael Rowan
http://queue.acm.org/detail.cfm?id=1831329

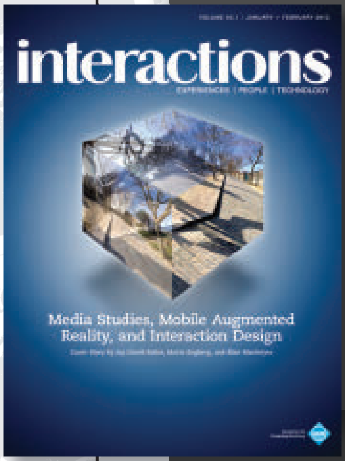
References

1. Barron, D.W. (Ed.). *Pascal–The Language and its Implementation.* John Wiley, 1981.
2. Common Intermediate Language (CIL); http://en.wikipedia.org/wiki/Common_Intermediate_Language.
3. CUDA; http://www.nvidia.com/object/cuda_home_new.html.
4. HMPP; http://www.caps-entreprise.com/openhmpp-directives/.
5. HSA Foundation; http://www.hsafoundation.com/.
6. Java bytecode; http://www.javaworld.com/jw-09-1996/jw-09-bytecodes.html.
7. JVM languages; http://en.wikipedia.org/wiki/List_of_JVM_languages.
8. OpenCL; http://www.khronos.org/opencl/.
9. Open source compilers; http://en.wikipedia.org/wiki/List_of_compilers#Open_source_compilers.

Fred Chow (chowfred@icubecorp.com) pioneered the first optimizing compiler for RISC processors, the MIPS Ucode compiler. He was the chief architect behind the Pro64 compiler at SGI, later open sourced as the Open64 compiler. He later created the widely accepted PathScale version of the Open64 compiler. He is currently leading the compiler effort for a new processor at ICube Corp.

interactions

EXPERIENCES | PEOPLE | TECHNOLOGY


interactions
Media Studies, Mobile Augmented Reality, and Interaction Design

interactions' website interactions.acm.org, is designed to capture the influential voice of its print component in covering the fields that envelop the study of people and computers.

The site offers a rich history of the conversations, collaborations, and discoveries from issues past, present, and future.

Check out the current issue, follow our bloggers, look up a past prototype, or discuss an upcoming trend in the communities of design and human-computer interaction.

FEATURES

BLOGS

FORUMS

DOWNLOADS

interactions.acm.org

Association for Computing Machinery

DOI:10.1145/2500499

Big data promises automated actionable knowledge creation and predictive models for use by both humans and computers.

BY VASANT DHAR

Data Science and Prediction

USE OF THE term "data science" is increasingly common, as is "big data." But what does it mean? Is there something unique about it? What skills do "data scientists" need to be productive in a world deluged by data? What are the implications for scientific inquiry? Here, I address these questions from the perspective of predictive modeling.

The term "science" implies knowledge gained through systematic study. In one definition, it is a systematic enterprise that builds and organizes knowledge in the form of testable explanations and predictions.[11] Data science might therefore imply a focus involving data and, by extension, statistics, or the systematic study of the organization, properties, and analysis of data and its role in inference, including our confidence in the inference. Why then do we need a new term like data science when we have had statistics for centuries? The fact that we now have huge amounts of data should not in and of itself justify the need for a new term.

The short answer is data science is different from statistics and other existing disciplines in several important ways. To start, the raw material, the "data" part of data science, is increasingly heterogeneous and unstructured—text, images, video—often emanating from networks with complex relationships between their entities. Figure 1 outlines the relative expected volumes of unstructured and structured data from 2008 to 2015 worldwide, projecting a difference of almost 200 petabytes (PB) in 2015 compared to a difference of 50PB in 2012. Analysis, including the combination of the two types of data, requires integration, interpretation, and sense making that is increasingly derived through tools from computer science, linguistics, econometrics, sociology, and other disciplines. The proliferation of markup languages and tags is designed to let computers interpret data automatically, making them active agents in the process of decision making. Unlike early markup languages (such as HTML) that emphasized the display of information for human consumption, most data generated by humans and computers today is for consumption by computers; that is, computers increasingly do background work for each other and make decisions automatically. This scalability in decision making has become possible because of big data that serves as the raw material for the creation of new knowledge; Watson, IBM's "Jeopardy!" champion, is a prime illustration of an emerging machine intelligence fueled by data and state-of-the-art analytics.

» key insights

- **Data science is the study of the generalizable extraction of knowledge from data.**
- **A common epistemic requirement in assessing whether new knowledge is actionable for decision making is its predictive power, not just its ability to explain the past.**
- **A data scientist requires an integrated skill set spanning mathematics, machine learning, artificial intelligence, statistics, databases, and optimization, along with a deep understanding of the craft of problem formulation to engineer effective solutions.**

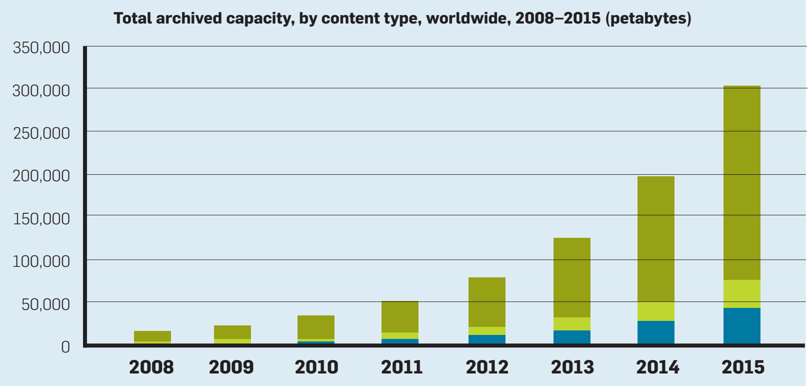
ansition to the deficit period
Repeated recessions
Surplus
Deficit
January
February
March
April
600
500
400
300
200
100
0

From an engineering perspective, scale matters in that it renders the traditional database models somewhat inadequate for knowledge discovery. Traditional database methods are not suited for knowledge discovery because they are optimized for fast access and summarization of data, given what the user wants to ask, or a query, not discovery of patterns in massive swaths of data when users lack a well-formulated query. Unlike database querying, which asks "What data satisfies this pattern (query)?" discovery asks "What patterns satisfy this data?" Specifically, our concern is finding interesting and robust patterns that satisfy the data, where "interesting" is usually something unexpected and actionable and "robust" is a pattern expected to occur in the future.

What makes an insight actionable? Other than domain-specific reasons, it is its predictive power; the return distribution associated with an action can be reliably estimated from past data and therefore acted upon with a high degree of confidence.

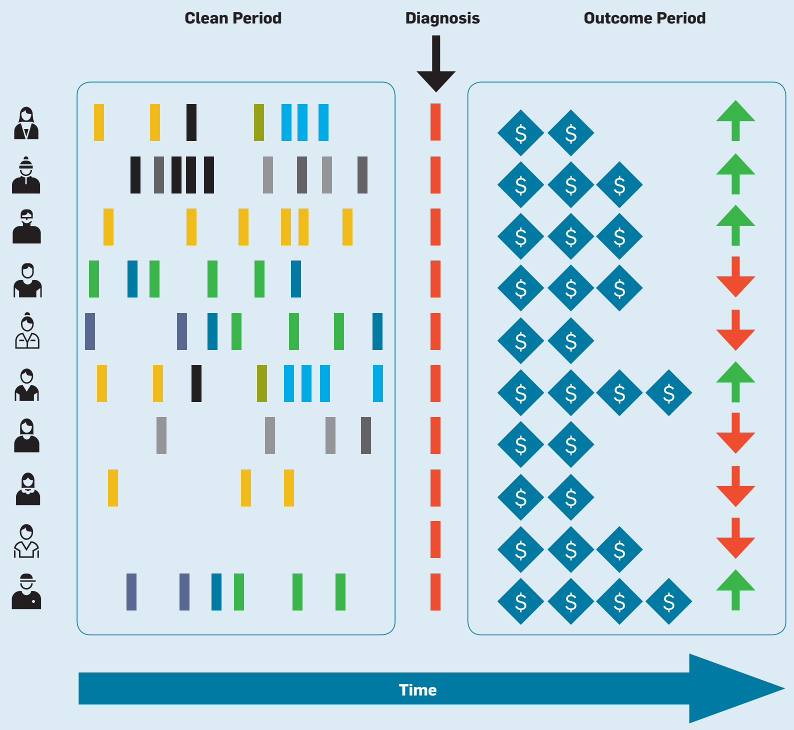
The emphasis on prediction is particularly strong in the machine learning and knowledge discovery in databases, or KDD, communities. Unless a learned model is predictive, it is generally regarded with skepticism, a position mirroring the view expressed by the 20th-century Austro-British philosopher Karl Popper as a primary criterion for evaluating a theory and for scientific progress in general.[24] Popper argued that theories that sought only to explain a phenomenon were weak, whereas those that made "bold predictions" that stand the test of time despite being readily falsifiable should be taken more seriously. In his well-known 1963 treatise on this subject, *Conjectures and Refutations*, Popper characterized Albert Einstein's theory of relativity as a "good" one since it made bold predictions that could be falsified; all attempts at falsification of the theory have indeed failed. In contrast, Popper argued that theories of psychoanalyst pioneers Sigmund Freud and Alfred Adler could be "bent" to accommodate virtually polar opposite scenarios and are weak in that they are virtually unfalsifiable.[a] The emphasis on predictive accuracy implicitly favors "simple" theories over more complex theories in that the accuracy of sparser models tends to be more robust on future data.[4,20] The requirement on predictive accuracy on observations that

Figure 1. Projected growth of unstructured and structured data.



	2008	2009	2010	2011	2012	2013	2014	2015
🟩 Unstructured	11,430	16,737	25,127	39,237	59,600	92,536	147,885	226,716
🟩 Database	1,952	2,782	4,065	6,179	9,140	13,824	21,532	32,188
🟦 Email	1,652	2,552	4,025	6,575	10,411	16,796	27,817	44,091

Figure 2. Health-care-use database snippet.



a Popper used opposite cases of a man who pushes a child into water with the intention of drowning the child and that of a man who sacrifices his life in an attempt to save the child. In Adler's view, the first man suffered from feelings of inferiority (producing perhaps the need to prove to himself that he dared to commit the crime), and so did the second man (whose need was to prove to himself that he dared to rescue the child at the expense of his own life).

will occur in the future is a key consideration in data science.

In the rest of this article, I cover the implications of data science from a business and research standpoint, first for skills, or what people in industry need to know and why. How should educators think about designing programs to deliver the skills most efficiently and enjoyably? And what kinds of decision-making skills will be required in the era of big data and how will they differ from the past when data was less plentiful?

The second part of my answer to defining big-data skills is aimed at research. How can scientists exploit the abundance of data and massive computational power to their advantage in scientific inquiry? How does this new line of thinking complement traditional methods of scientific inquiry? And how can it augment the way data scientists think about discovery and innovation?

It is not uncommon for two experts in the social sciences to propose opposite relationships among the variables and offer diametrically opposite predictions based on the same sets of facts.

Implications

A 2011 McKinsey industry report[19] said the volume of data worldwide is growing at a rate of approximately 50% per year, or a roughly 40-fold increase since 2001. Hundreds of billions of messages are transmitted through social media daily and millions of videos uploaded daily across the Internet. As storage becomes almost free, most of it is stored because businesses generally associate a positive option value with data; that is, since it may turn out to be useful in ways not yet foreseen, why not just keep it? (One indicator of how inexpensive storage is today is the fact that it is possible to store the world's entire stock of music on a $500 device.)

Using large amounts of data for decision making became practical in the 1980s. The field of data mining burgeoned in the early 1990s as relational database technology matured and business processes were increasingly automated. Early books on data mining[6,7,17] from the 1990s described how various methods from machine learning could be applied to a variety of business problems. A corresponding explosion involved software tools geared toward leveraging transactional and behavioral data for purposes of explanation and prediction.

An important lesson learned in the 1990s is that machine learning "works" in the sense that these methods detect subtle structure in data relatively easily without having to make strong assumptions about linearity, monotonicity, or parameters of distributions. The downside of these methods is they also pick up the noise in data,[31] often with no way to distinguish between signal and noise, a point I return to shortly.

Despite their drawbacks, a lot can be said for methods that do not force us to make assumptions about the nature of the relationship between variables before we begin our inquiry. This is not trivial. Most of us are trained to believe theory must originate in the human mind based on prior theory, with data then gathered to demonstrate the validity of the theory. Machine learning turns this process around. Given a large trove of data, the computer taunts us by saying, "If only you knew what question to ask me, I would give you some very interesting answers based on the data." Such a capability is powerful since we often do not know what question to ask. For example, consider a health-care database of individuals who have been using the health-care system for many years, where among them a group has been diagnosed with Type 2 diabetes, and some subset of this group has developed complications. It could be very useful to know whether there are any patterns to the complications and whether the probability of complications can be predicted and therefore acted upon. However, it is difficult to know what specific query, if any, might reveal such patterns.

To make this scenario more concrete, consider the data emanating from a health-care system that essentially consists of "transactions," or points of contact over time between a patient and the system. Records include services rendered by health-care providers or medication dispensed on a particular date; notes and observations could also be part of the record. Figure 2 outlines what the raw data would look like for 10 individuals where the data is separated into a "clean period" (history prior to diagnosis), a red bar ("diagnosis"), and the "outcome period" (costs and other outcomes, including complications). Each colored bar in the clean period represents a medication, showing the first individual was on seven different medications prior to diagnosis, the second on nine, the third on six, and so on. The sixth and tenth individuals were the costliest to treat and developed complications, as did the first three, represented by the upward-pointing green arrows.

Extracting interesting patterns is nontrivial, even from a tiny temporal database like this. Are complications associated with the yellow meds or with the gray meds? The yellows in the absence of the blues? Or is it more than three yellows or three blues? The list goes on. Even more significant, perhaps if we created "useful" features or aggregations from the raw data, could physicians, insurers, or policy makers predict likely complications for individuals or for groups of people?

Feature construction is an important creative step in knowledge discovery. The raw data across individuals typically needs to be aggregated into some sort of canonical form before useful patterns can be discovered; for example, suppose we could count the number of prescriptions an individual is on without regard to the specifics of each prescription as one approximation of the "health status" of the individual prior to diagnosis. Such a feature ignores the "severity" or other characteristics of the individual medications, but such aggregation is nonetheless typical of feature engineering.

Suppose, too, a "complications database" would be synthesized from the data, possibly including demographic information (such as patient age and medical history); it could also include health status based on a count of current medications; see Figure 3, in which a learning algorithm, designated by the right-facing blue arrow, could be applied to discover the pattern on the right. The pattern represents an abstraction of the data, or the type of question we should ask the database, if only we knew what to ask. Other data transformations and aggregations could yield other medically insightful patterns.

What makes the pattern on the right side of Figure 3 interesting? Suppose the overall complication rate in the population is 5%; that is, a random sample of the database includes, on average, 5% complications. In this scenario, the snippet on the right side of Figure 3 could be very interesting since its complication rate is many times greater than the average. The critical question is whether this is a pattern that is robust and hence predictive, likely to hold up in other cases in the future. The issue of determining robustness has been addressed extensively in the machine learning literature and is a key consideration for data scientists.[23]

A new powerful method is available for theory development not previously practical due to the paucity of data.

If Figure 3 is representative of the larger database, the box on the right tells us the interesting question to ask the database: "What is the incidence of complications in Type 2 diabetes for people over age 36 who are on six or more medications?" In terms of actionability, such a pattern might suggest being extra vigilant about people with such a profile who do not currently have a complication in light of their high susceptibility to complications.

The general point is that when data is large and multidimensional, it is practically impossible for us to know a priori that a query (such as the one here concerning patterns in diabetes complications) is a good one, or one that provides a potentially interesting and actionable insight. Suitably designed machine learning algorithms help find such patterns for us. To be useful both practically and scientifically, the patterns must be predictive. The emphasis on predictability typically favors Occam's razor, or succinctness, since simpler models are more likely to hold up on future observations than more complex ones, all else being equal;[4] for example, consider the diabetes complication pattern here:

Age > 36 and #Medication >
6 $\longrightarrow$ Complication_rate=100%

A simpler competing model might ignore age altogether, stating simply that people on six or more medications tend to develop complications. The reliability of such a model would be more apparent when applied to future data; for example, does simplicity lead to greater future predictive accuracy in terms of fewer false positives and false negatives? If it does, it is favored. The practice of "out of sample" and "out of time" testing is used by data scientists to assess the robustness of patterns from a predictive standpoint.

When predictive accuracy is a primary objective in domains involving massive amounts of data, the computer tends to play a significant role in model building and decision making. The computer itself can build predictive models through an intelligent "generate and test" process, with the end result an assembled model that is the decision maker; that is, it automates Popper's criterion of predictive accuracy for evaluating models at a scale in ways not feasible before.

If we consider one of these patterns—that people with "poor health status" (proxied by number of medications) have high rates of complications—can we say poor health status "causes" complications? If so, perhaps we can intervene and influence the outcome by possibly controlling the number of medications. The answer is: it depends. It could be the case that the real cause is not in our observed set of variables. If we assume we have observed all relevant variables that could be causing complications, algorithms are available for extracting causal structure from data,[21] depending how the data was generated. Specifically, we still need a clear understanding of the "story" behind the data in order to know whether the possibility of causation can and should be entertained, even in principle. In our example of patients over age 36 with Type 2 diabetes, for instance, was it the case that the people on seven or more medications were "inherently sicker" and would have developed complications anyway? If so, it might be incorrect to conclude that large numbers of medications cause complications. If, on the other hand, the observational data followed a "natural experiment" where treatments were assigned randomly to comparable individuals and enough data is available for calculating the relevant conditional probabilities, it might be feasible to extract a causal model that could be used for intervention. This issue of extracting a causal model from data is addressed in the following sections; for a more complete treatment on causal models, see Pearl,[21] Sloman,[29] and Spirtes et al.[30]

Skills

Machine learning skills are fast becoming necessary for data scientists as companies navigate the data deluge and try to build automated decision systems that hinge on predictive accuracy.[25] A basic course in machine learning is necessary in today's marketplace. In addition, knowledge of text processing and "text mining" is becoming essential in light of the explosion of text and other unstructured data in healthcare systems, social networks, and other forums. Knowledge about markup languages like XML and its derivatives is also essential, as content becomes tagged and hence able to be interpreted automatically by computers.

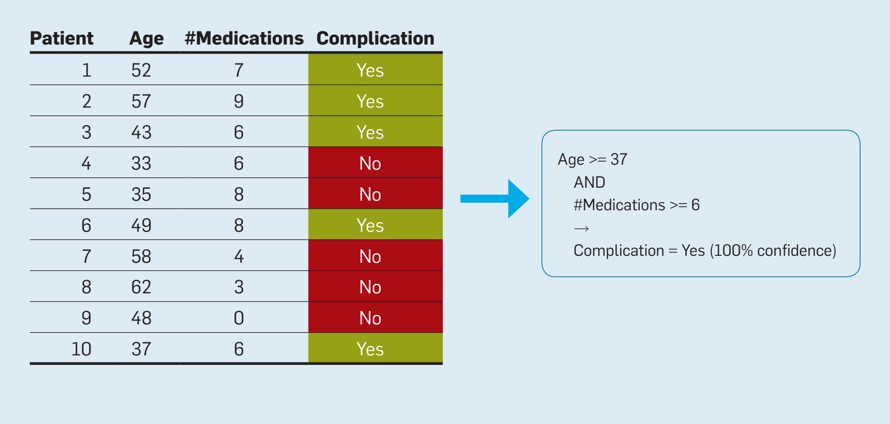
Data scientists' knowledge about machine learning must build on more basic skills that fall into three broad classes: The first is statistics, especially Bayesian statistics, which requires a working knowledge of probability, distributions, hypothesis testing, and multivariate analysis. It can be acquired in a two- or three-course sequence. Multivariate analysis often overlaps with econometrics, which is concerned with fitting robust statistical models to economic data. Unlike machine learning methods, which make no or few assumptions about the functional form of relationships among variables, multivariate analysis and econometrics by and large focus on estimating parameters of linear models where the relationship between the dependent and independent variables is expressed as a linear equality.

The second class of skills comes from computer science and pertains to how data is internally represented and manipulated by computers. This involves a sequence of courses on data structures, algorithms, and systems, including distributed computing, databases, parallel computing, and fault-tolerant computing. Together with scripting languages (such as Python and Perl), systems skills are the fundamental building blocks required for dealing with reasonable-size datasets. For handling very large datasets, however, standard database systems built on the relational data model have severe limitations. The recent move toward cloud computing and nonrelational structures for dealing with enormous datasets in a robust manner signals a new set of required skills for data scientists.

The third class of skills requires knowledge about correlation and causation and is at the heart of virtually any modeling exercise involving data. While observational data generally limits us to correlations, we can get lucky. Sometimes plentiful data might represent natural randomized trials and the possibility of calculating conditional probabilities reliably, enabling discovery of causal structure.[22] Building causal models is desirable in domains where one has reasonable confidence as to the completeness of the formulated model and its stability, or whether the causal model "generating" the observed data is stable. At the very least, a data scientist should have a clear idea of the distinction between correlation and causality and the ability to assess which models are feasible, desirable, and practical in different settings.

The final skill set is the least standardized and somewhat elusive and to

Figure 3. Extracting interesting patterns in health outcomes from health-care system use.



Patient	Age	#Medications	Complication
1	52	7	Yes
2	57	9	Yes
3	43	6	Yes
4	33	6	No
5	35	8	No
6	49	8	Yes
7	58	4	No
8	62	3	No
9	48	0	No
10	37	6	Yes

Age >= 37
AND
#Medications >= 6
→
Complication = Yes (100% confidence)

some extent a craft but also a key differentiator to be an effective data scientist—the ability to formulate problems in a way that results in effective solutions. Herbert Simon, the 20th-century American economist who coined the term "artificial intelligence" demonstrated that many seemingly different problems are often "isomorphic," or have the identical underlying structure. He demonstrated that many recursive problems could be expressed as the standard Towers of Hanoi problem, or involving identical initial and goal states and operators. His larger point was it is easy to solve seemingly difficult problems if represented creatively with isomorphism in mind.[28]

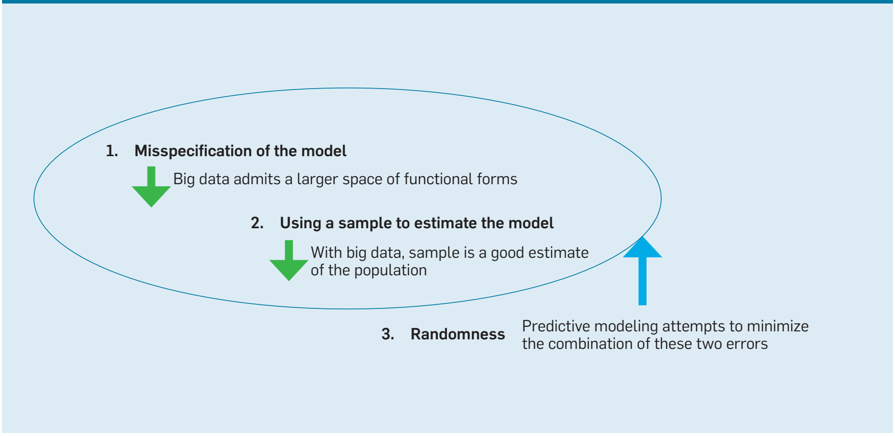
In a broader sense, formulation expertise involves the ability to see commonalities across very different problems; for example, many problems have "unbalanced target classes" usually denoting the dependent variable is interesting only sometimes (such as when people develop diabetes complications or respond to marketing offers or promotions). These are the cases of interest we would like to predict. Such problems are a challenge for models that, in Popperian terms, must go out on a limb to make predictions that are likely to be wrong unless the model is extremely good at discriminating among the classes. Experienced data scientists are familiar with these problems and know how to formulate them in a way that gives a system a chance to make correct predictions under conditions where the priors are stacked heavily against it.

Problem-formulation skills represent core skills for data scientists over the next decade. The term "computational thinking" coined by Papert[21] and elaborated by Wing[32] is similar in spirit to the skills described here. There is considerable activity in universities to train students in problem-formulation skills and provide electives structured around the core that are more suited to specific disciplines.

The data science revolution also poses serious organizational challenges as to how organizations manage their data scientists. Besides recognizing and nurturing the appropriate skill sets, it requires a shift in managers' mind-sets toward data-driven decision making to replace or augment intuition and past practices. A famous quote by 20th-century American statistician W. Edwards Demming—"In God we trust, everyone else please bring data"—has come to characterize the new orientation, from intuition-based decision making to fact-based decision making.

From a decision-making standpoint, we are moving into an era of big data where for many types of problems computers are inherently better decision makers than humans, where "better" could be defined in terms of cost, accuracy, and scalability. This shift has already happened in the world of data-intensive finance where computers make the majority of investment decisions, often in fractions of a second, as new information becomes available. The same holds in areas of online advertising where millions of auctions are conducted in milliseconds every day, air traffic control, routing of package delivery, and many types of planning tasks that require scale, speed, and accuracy simultaneously, a trend likely to accelerate in the next few years.

Figure 4. Sources of error in predictive models and their mitigation.


1. Misspecification of the model
Big data admits a larger space of functional forms
2. Using a sample to estimate the model
With big data, sample is a good estimate of the population
3. Randomness
Predictive modeling attempts to minimize the combination of these two errors

Knowledge Discovery

Former editor of *Wired* magazine Chris Anderson[1] drew on the quote by British-born statistician George Box that "All models are wrong, but some are useful," arguing, with the huge amounts of data available today, we do not need to settle for wrong models or any models at all. Anderson said prediction is of paramount importance to businesses, and data can be used to let such models emerge through machine learning algorithms, largely unaided by humans, pointing to companies like Google as symbolizing the triumph of machine learning over top-down theory development. Google's language translator does not "understand" language, nor do its algorithms know the contents of webpages. IBM's Watson does not "understand" the questions it is asked or use deep causal knowledge to generate questions to the answers it is given. There are dozens of lesser-known companies that likewise are able to predict the odds of someone responding to a display ad without a solid theory but rather based on gobs of data about the behavior of individuals and the similarities and differences in that behavior.

Anderson's 2008 article launched a vigorous debate in academic circles. How can one have science and predictive models without first articulating a theory?

The observation by Dhar and Chou[5] that "patterns emerge before reasons for them become apparent" tends to resonate universally among professionals, particularly in financial markets, marketing, health care, and fields that study human behavior. If this is true, Box's observation becomes relevant: If a problem is nonstationary and a model is only an approximation anyway, why not build the best predictive model based on data available until that time and just update it periodically? Why bother developing a detailed causal model if it is poor at prediction and, more important, likely to get worse over time due to "concept drift"?

Some scientists would say there is no theory without causality, that all observational data, except total chaos, must be generated from a causal model. In the earlier health-care example involving medical complications in patients with Type 2 diabetes, this seems obvious; some underlying mechanism must have been responsible for the observed outcomes. But we may not have observed or been capable of observing the causal picture. Even if we observed the right variables we would need to know how the observational data was generated before we can in principle draw causal connections. If the observations represent a natural experiment (such as physicians using a new drug vs. other physicians using an old one for comparable individuals), the data might reveal causality. On the other hand, if the new drug is prescribed primarily for "sicker" individuals, it would represent a specific kind of bias in the data.

Anderson's point has particular relevance in the health, social, and earth sciences in the era of big data since these areas are generally characterized by a lack of solid theory but where we now see huge amounts of data that can serve as grist for theory building[3,12,13] or understanding large-scale social behavior and attitudes and how they can be altered.[14] Contrast physics and social sciences at opposite ends of the spectrum in terms of the predictive power of their theories. In physics, a theory is expected to be "complete" in the sense a relationship among certain variables is intended to explain the phenomenon completely, with no exceptions. Such a model is expected to make perfect predictions—subject to measurement error but not to error due to omitted variables or unintended consequences. In such domains, the explanatory and predictive models are synonymous. The behavior of a space shuttle is, for example, explained completely by the causal model describing the physical forces acting on it. This model can also be used to predict what will happen if any input changes. It is not sufficient to have a model 95% sure of outcomes and leave the rest to chance. Engineering follows science.

In contrast, the social sciences are generally characterized by incomplete models intended to be partial approximations of reality, often based on assumptions of human behavior known to be simplistic. A model correct 95% of the time in this world would be considered quite good. Ironically, however, the emphasis in social science

Big data makes it feasible for a machine to ask and validate interesting questions humans might not consider.

theory development is often on proposing theories that embody causality without serious consideration of their predictive power. When such a theory claims "A causes B," data is gathered to confirm whether the relationship is indeed causal. But its predictive accuracy could be poor because the theory is incomplete. Indeed, it is not uncommon for two experts in the social sciences to propose opposite relationships among the variables and offer diametrically opposite predictions based on the same sets of facts; for example, economists routinely disagree on both theory and prediction, and error rates of forecasts tend to be high.

How could big data put these domains on firmer ground? In the "hard" sciences, where models can be assumed, for practical purposes, to be complete, there exists the possibility of extracting causal models from large amounts of data. In other fields, large amounts of data can result in accurate predictive models, even though no causal insights are immediately apparent. As long as their prediction errors are small, they could still point us in the right direction for theory development. As an example of being pointed in the right direction, a health-care research scientist recently remarked on an observed pattern of coronary failure being preceded months earlier by a serious infection. One of his conjectures was infections might have caused inflamed arteries and loosened plaque that subsequently caused coronary failure. There could be other explanations, but if the observed pattern is predictive, it might be worthy of publication and deeper inquiry. The questions such a case raise for gatekeepers of science is whether to more strongly consider the Popperian test of predictive accuracy on future data and favor simple accurate predictive models as potential components of future theory instead of requiring a causal model up front tested by the data.

What makes predictive models accurate? Conversely, where do errors come from?

Hastie et al.[10] said errors in prediction come from three sources: The first is misspecification of a model, so, for example, a linear model that attempts to fit a nonlinear phenomenon could generate an error simply because the linear model imposes an inappropriate bias on the problem. The second is the samples used for estimating parameters; the smaller the samples, the greater the bias in the model's estimates. And the third is randomness, even when the model is specified perfectly.

Big data allows data scientists to significantly reduce the first two types of error (see Figure 4). Large amounts of data allow us to consider models that make fewer assumptions about functional form than linear or logistic regressions simply because there is a lot more data to test such models and compute reliable error bounds.[27] Big data also eliminates the second type of error, as sample estimates become reasonable proxies for the population.

The theoretical limitation of observational data of the sort in these examples, regardless of how big it is, is that the data is generally "passive," representing what actually happened in contrast to the multitude of things that could have happened had circumstances been different. In health care, it is like having observed the use of the health-care system passively and now having the chance of understand it in retrospect and extract predictive patterns from it. Unless we are fortunate enough that the data provided us the right experiments naturally, it does not tell us what could have happened if some other treatment had been administered to a specific patient or to an identical patient; that is, it does not represent a clean controlled randomized experiment where the researcher is able to establish controls and measure the differential effect of treatments on matched pairs.

Interestingly, however, the Internet era is fertile ground for conducting inexpensive large-scale randomized experiments on social behavior; Kohavi et al.[15] provide a number of examples. A 2012 controlled experiment by Aral and Walker[2] on the adoption of video games asked whether it was "influence" or "homophily" that affected choice uncovered profiles of people who are influential and susceptible. Results include patterns (such as "older men are more influential than younger men" and "people of the same age group have more influence on each other than from other age groups"). While specific to games, these results suggest influence is nuanced, certainly more so than existing theories like Malcolm Gladwell's concept of "super influencers"[8] and myriad other popular theories. Big data provides a basis for testing them.

Predictive modeling and machine learning are increasingly central to the business models of Internet-based data-driven businesses.

One of the most far-reaching modern applications of big data is in politics, as exemplified by the Democratic National Committee heavy investment in data and analytics prior to President Barack Obama's winning 2012 campaign, debunking widely held beliefs (such as voters in the "middle" are most critical to outcomes, when in fact issues that resonate with some segments of solidly partisan voters can sway them[14]). In the campaign, the DNC crafted predictive models on the basis of results from large-scale experiments used to manipulate attitudes. The campaign predicted at the level of individual voters how each eligible voter would vote, as well as how to "turn someone into the type of person it wanted you to be."[14]

Social science theory building is also likely to get a good boost from big data and machine learning. Never before have social scientists been able to observe human behavior at the degree of granularity and variability seen today with increasing amounts of human interaction and economic activity mediated by the Internet. While the inductive method has limitations, the sheer volume of data being generated makes induction not only feasible but productive. That is not to say the traditional scientific method is "dead," as claimed by Anderson.[1] On the contrary, it continues to serve us well. However, a new powerful method is available for theory development not previously practical due to the paucity of data. That era of limited data and its associated assumptions is largely over.

Conclusion

Hypothesis-driven research and approaches to theory development have served us well. But a lot of data is emanating around us where these traditional approaches to identifying structure do not scale well or take advantage of observations that would not occur under controlled circumstances; for example, in health care, controlled experiments have helped identify many causes of disease but may not reflect the actual complexities of health.[3,18] Indeed, some estimates claim clinical trials exclude as much as 80% of the situations in which a drug might be prescribed, as when a patient is on multiple medications.[3] In situations where we are able to design randomized trials, big data makes it feasible to uncover the causal models generating the data.

As shown earlier in the diabetes-related health-care example, big data makes it feasible for a machine to ask and validate interesting questions humans might not consider. This capability is indeed the foundation for building predictive modeling, which is key to actionable business decision making. For many data-starved areas of inquiry, especially health care and the social, ecological, and earth sciences, data provides an unprecedented opportunity for knowledge discovery and theory development. Never before have these areas had data of the variety and scale available today.

This emerging landscape calls for the integrative skill set identified here as essential for emerging data scientists. Academic programs in computer science, engineering, and business management teach a subset of these skills but have yet to teach the integration of skills needed to function as a data scientist or to manage data scientists productively. Universities are scrambling to address the lacunae and provide a more integrated skill set covering basic skills in computer science, statistics, causal modeling, problem isomorphs and formulation, and computational thinking.

Predictive modeling and machine learning are increasingly central to the business models of Internet-based data-driven businesses. An early success, Paypal, was able to capture and dominate consumer-to-consumer payments due to its ability to predict the distribution of losses for each transaction and act accordingly. This data-driven ability was in sharp contrast to the prevailing practice of treating transactions identically from a risk standpoint. Predictive modeling is also at the heart of Google's search engine and several other products. But the first machine that could arguably be considered to pass the Turing test and create new insights in the course of problem solving is IBM's Watson, which makes extensive use of learning and prediction in its problem-solving process. In a game like "Jeopardy!," where understanding the question itself is often nontrivial and the domain open-ended and nonstationary, it is not practical to be successful through an extensive enumeration of possibilities or top-down theory building. The solution is to endow a computer with the ability to train itself automatically based on large numbers of examples. Watson also demonstrated the power of machine learning is greatly amplified through the availability of high-quality human-curated data, as in Wikipedia. This trend—combining human knowledge with machine learning—also appears to be on the rise. Google's recent foray in the Knowledge Graph[16] is intended to enable the system to understand the entities corresponding to the torrent of strings it processes continuously. Google wants to understand "things," not just "strings."[26]

Organizations and managers face significant challenges in adapting to the new world of data. It is suddenly possible to test many of their established intuitions, experiment cheaply and accurately, and base decisions on data. This opportunity requires a fundamental shift in organizational culture, one seen in organizations that have embraced the emerging world of data for decision making. **C**

References

1. Anderson, C. The end of theory: The data deluge makes the scientific method obsolete. *Wired 16*, 7 (June 23, 2008).
2. Aral, S. and Walker, D. Identifying influential and susceptible members of social networks. *Science 337*, 6092 (June 21, 2012).
3. Buchan, I., Winn, J., and Bishop, C. *A Unified Modeling Approach to Data-Intensive Healthcare. The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, WA, 2009.
4. Dhar, V. Prediction in financial markets: The case for small disjuncts. *ACM Transactions on Intelligent Systems and Technologies 2*, 3 (Apr. 2011).
5. Dhar, V. and Chou, D. A comparison of nonlinear models for financial prediction. *IEEE Transactions on Neural Networks 12*, 4 (June 2001), 907–921.
6. Dhar, V. and Stein, R. *Seven Methods for Transforming Corporate Data Into Business Intelligence.* Prentice-Hall, Englewood Cliffs, NJ, 1997.
7. Frawley, W. and Piatetsky-Shapiro, G., Eds. *Knowledge Discovery in Databases.* AAAI/MIT Press, Cambridge, MA, 1991.
8. Gladwell, M. *The Tipping Point: How Little Things Can Make a Big Difference.* Little Brown, New York, 2000.
9. Goel, S., Watts, D., and Goldstein, D. The structure of online diffusion networks. In *Proceedings of the 13th ACM Conference on Electronic Commerce* (2012), 623–638.
10. Hastie, T., Tibsharani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, 2009.
11. Heilbron, J.L., Ed. *The Oxford Companion to the History of Modern Science.* Oxford University Press, New York, 2003.
12. Hey, T., Tansley, S., and Tolle, K., Eds. 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, WA, 2009.
13. Hunt, J., Baldochi, D., and van Ingen, C. *Redefining Ecological Science Using Data. The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, WA, 2009.
14. Issenberg, S. A more perfect union: How President Obama's campaign used big data to rally individual voters. *MIT Technology Review* (Dec. 2012).
15. Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. Controlled experiments on the Web: Survey and practical guide. *Data Mining and Knowledge Discovery 18* (2009), 140–181.
16. Lin, T., Patrick, P., Gamon, M., Kannan, A., and Fuxman, A. Active objects: Actions for entity-centric search. In *Proceedings of the 21st International Conference on the World Wide Web* (Lyon, France). ACM Press, New York, 2012.
17. Linoff, G. and Berry, M. *Data Mining Techniques: For Marketing, Sales, and Customer Support.* John Wiley & Sons, Inc., New York, 1997.
18. Maguire, J. and Dhar, V. Comparative effectiveness for oral anti-diabetic treatments among newly diagnosed Type 2 diabetics: Data-driven predictive analytics in healthcare. *Health Systems 2* (2013), 73–92.
19. McKinsey Global Institute. *Big Data: The Next Frontier for Innovation, Competition, and Productivity.* Technical Report, June 2011.
20. Meinshausen, N. Relaxed lasso. *Computational Statistics & Data Analysis 52*, 1 (Sept. 15, 2007), 374–393.
21. Papert, S. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning 1*, 1 (1996), 95–123.
22. Pearl, J. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, Cambridge, U.K., 2000.
23. Perlich, C., Provost, F., and Simonoff, J. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research 4*, 12 (2003), 211–255.
24. Popper, K. *Conjectures and Refutations.* Routledge, London, 1963.
25. Provost, F. and Fawcett, T. *Data Science for Business.* O'Reilly Media, New York, 2013.
26. Roush, W. Google gets a second brain, changing everything about search. *Xconomy* (Dec. 12, 2012); http://www.xconomy.com /san-francisco/2012/12/12/google-gets-a-second-brain-changing-everything-about-search/?single_page=true
27. Shmueli, G. To explain or to predict? *Statistical Science 25*, 3 (Aug. 2010), 289–310.
28. Simon, H.A. and Hayes, J.R. The understanding process: Problem isomorphs. *Cognitive Psychology 8*, 2 (Apr. 1976), 165–190.
29. Sloman, S. *Causal Models.* Oxford University Press, Oxford, U.K. 2005.
30. Spirtes, P., Scheines, R., and Glymour, C. *Causation, Prediction and Search.* Springer, New York, 1993.
31. Tukey, J.W. *Exploratory Data Analysis.* Addison-Wesley, Boston, 1977.
32. Wing, J. Computational thinking. *Commun. ACM 49*, 3 (Mar. 2006), 33–35.

Vasant Dhar (vdhar@stern.nyu.edu) is a professor and co-director of the Center for Business Analytics at the Stern School of Business at New York University, New York.

DOI:10.1145/2500498

Accessible information technology is not just good design and a clever way to win new users, it is the law.

BY JONATHAN LAZAR AND HARRY HOCHHEISER

Legal Aspects of Interface Accessibility in the U.S.

ASKING "WHY IS accessibility so hard?," Vinton G. Cerf explored some of the design challenges associated with building interfaces for users with disabilities.[2] As real as these difficulties may be, they fail to tell the whole story. The technical complexity of making interfaces accessible to people with visual, auditory, motor, or cognitive impairments is matched by a daunting regulatory and legal framework. In the U.S., accessibility is the subject of the numerous federal statutes, regulations, and reports that define implementation of complex legislation. A growing body of legal precedents, as well as state and local laws, adds further complexity.

Wading through it all might cause some software developers to long for the simplicity of building cross-browser websites. For example, Section 504 of the Rehabilitation Act and the Americans with Disabilities Act require universities to provide accessible information technology, though the technical standards defining compliance are in Section 508 of the Rehabilitation Act; for more on these laws, as well as other laws, regulations, specifications, and recommendations, see the sidebar "For More Information." State laws relating to accessible information technology add further complication. Inconsistencies between state and federal law, as well as a general lack of clear guidelines defining compliance with various laws, add more, particularly for software developers who generally lack formal training in the law concerning disability rights.

This complexity creates the perception that accessibility is somehow "hard" in a sense not discussed by Cerf. Faced with potentially inconsistent legal regimes, a multiplicity of standards and tools, and constantly advancing technologies, designers and developers might be at a loss. How can they meet user demands for the latest tools while still being legally compliant? Which of the hundreds of warnings from accessibility audit tools are worth paying attention to, and which are not? What is an acceptable level of accessibility? What multimedia must be captioned, and what need not be? How is legal compliance measured and monitored? Although the complaint "accessibility stifles innovation" is too often (and incorrectly) used to rationalize doing nothing, a legal landscape that makes for a more complex picture is to no one's advantage.

» key insights

- Although the complexity of interpreting multiple federal and state legal requirements can make accessibility design seem complicated, accessible design benefits all users.
- Understanding the implications of four key U.S. laws can help designers avoid compliance difficulties.
- Technical professionals, including ACM members, can help guide accessibility laws and regulations.

Computing professionals have played, and should continue to play, a leading role in achieving the goal of accessible information systems. This article takes up Cerf's challenge to the computing community to explore why accessibility is indeed so hard, examining the difficulties of accessibility from the perspective of technology and design, as well as complementary legal, regulatory, and policy perspectives. Although the focus is on U.S. public policy, the challenges and opportunities generalize to any country with laws related to information technology accessibility. We conclude with suggestions for how the computing community can engage in the policy arena to make accessibility less hard.

Accessible Design

Accessibility should inspire good design. Software and hardware are inherently flexible and adaptable, often to situations having nothing to do with accessibility. Screen readers that convert video display content to synthesized speech might also be used to, say, read a recipe out loud to a busy cook in a home kitchen. Written transcripts of online lectures might help students prepare for exams, even if they are able to hear the audio. Easy-to-use tools for adjustment of screen resolution and font size help tired-eyed users everywhere, not just those with poor vision. Appropriate markup, labeling, and captioning help users with disabilities while improving the accuracy of search engines. Accessibility features help users with disabilities, older users, and users with low-speed network connections in developing countries.

Accessible design need not be difficult. With more than 30 years of research experience, interface developers know how to make interfaces that work for people with perceptual and motor impairments. Research related to users with cognitive impairment is a newer but growing area. Interface developers have numerous guidelines available, not only from the U.S. Access Board (in the form of Section 508 technical specifications) but also from international technical bodies (such as the World Wide Web Consortium's Web Accessibility Initiative and the Web Content Accessibility Guidelines).

Accessibility also spurs innovation. Identifying creative solutions to challenging constraints is the essence of good engineering and design. Technologies once viewed as unique and only for people with disabilities are today built directly into operating systems. Everyone benefits. Apple's iOS (iPhone and iPad) operating systems include many novel accessibility features, including speech recognition, screen reading, and support for gesturing by people who might be unable to see or use a touchscreen. The challenge of providing greater accessibility has led to countless innovations across many areas of computing. Head-pointing devices and gesture-recognition systems developed for users with motor impairment have inspired widely used game interfaces. The need for contextual cues that help screen readers clearly interpret Web content led to the W3C's Accessible Rich Internet Applications (ARIA) recommendation. ARIA's tags describing the purpose of webpage design elements (such as menu and header) will be helpful to cooks, factory workers, and others who might want to access Web content without looking directly at a screen.

Accessibility should not be an afterthought. Just as interface experts say usability is more than adding a user interface to an otherwise completed system, designers and consumers of accessible tools and systems have learned that "adding on" accessibility features after the fact is ineffective. Incorporating screen readers, magnification software, audio output, and other accessibility features into commodity operating system software substantially reduces costs while simultaneously making these features available to users who are not necessarily disabled. However, such features do not work properly with software and websites if the software and websites are not properly coded through existing standards, with labels and tags necessary for the assistive technologies.

Uncertainty as to which accessibility requirements must be met under which circumstances leaves developers in a quandary: When are their products, software, and websites accessible "enough"?

Evolution of accessible Web design demonstrates the value of this approach. Early approaches to accessible Web design, dating from the mid-1990s, before release of the Web Content Accessibility Guidelines 1.0 recommendation, focused on so-called "text-only" pages that repurposed existing content, providing accessibility at high cost in terms of maintenance of parallel sites (often not kept up to date). Today's best practices involving Web standards focus on Web layout techniques, good code documentation, and Web design conventions to support accessible design with little additional maintenance.

Accepted best practices in software design can help encourage accessibility. Standards, guidelines, and conventions support both accessibility and usability. Separating content from presentation helps developers decrease the cost of updating and internationalizing content while improving the performance of screen readers and other accessibility tools. Standard installations of Windows, OS X, and Linux have included accessibility features for more than 10 years, with availability of these features "out of the box," significantly simplifying matters for users. For developers, services provided by operating systems and development tools, including facilities for both creating and auditing accessibility features, can decrease perceived difficulty and encourage compliance.

Although some customized devices and designs may always be needed for users with extreme disabilities, the approaches covered here can go a long way toward achieving the goal of accessibility for most people with disabilities. However, tool support, technical arguments, and best intentions are not necessarily sufficient; appropriate legal frameworks are also needed for defining the roles and responsibilities of technologists and information providers.

Laws and Regulations

The complexities of accessibility law can be overwhelming for some interface designers, software engineers, Web developers, product designers, and others who focus primarily on technical, not legal, constraints. Accessibility laws come from multiple jurisdictions (in the U.S., primarily federal and state), varying in both scope and applicability, making interpretation a challenging task, at best.

Legislation is only a start. Even when the statutory legal landscape is reasonably stable, legal responsibilities could shift due to revisions to the regulations that define how laws are implemented and enforced, as well as to judicial rulings that clarify interpretation of the laws. Legal statutes with no corresponding implementation and enforcement are rarely effective. The most effective solutions generally combine clear legal statutes with policies and regulations for implementing them.

Although this dynamic landscape makes it difficult to make definitive statements, discussing U.S. legal requirements for accessibility in information technology can help provide a roadmap for understanding the big picture in IT accessibility law. Four major U.S. disability rights laws—Section 504 of the Rehabilitation Act (1973), Section 508 of the Rehabilitation Act (1998), the Americans With Disabilities Act (1990), and the 21st Century Communications and Video Accessibility Act (2010)—specifically relate to designing technology accessible for people with disabilities. Other domain-specific laws and regulations include the Individuals with Disabilities Education Act (IDEA) for K–12 education, the Help America Vote Act for voting rights, and the Nondiscrimination on the Basis of Disability in Air Travel regulation from the U.S. Department of Transportation.

Section 504 of the Rehabilitation Act (1973). As the first major disability rights legislation in the U.S, Section 504 requires recipients of federal funding to not discriminate against people with disabilities in their programs, services, benefits, or opportunities.[8] Its coverage is broad due to federal funding of institutions, including universities, health-care facilities, and federal contractors, as well as all federal and many state activities. Although Section 504 does not provide technical specifications for software engineers and developers, it is often cited as the legal reasoning behind civil actions.

Section 508 of the Rehabilitation Act (1998). Section 508 requires when "Federal agencies develop, procure, maintain, or use electronic and information technology" the technology is accessible to people with disabilities. Included is technology acquired or developed for federal employees, as well as technology available to the public (such as federal websites and kiosks). Section 508 technical standards have been available since 2000. The U.S. Department of Justice is responsible for biennial collection of data measuring Section 508 compliance.

Following a seven-year gap when the Department of Justice did not collect data on Section 508 compliance, a 2010 letter from the White House restarted 508 data collection and enforcement.[7] A September 2012 report from the Department of Justice reported the results of this effort, including the somewhat surprising finding that a number of federal agencies had failed to conform to Section 508 and had no plans to do so.[11] In a related effort, a White House invitation for online suggestions for improving Section 508 implementation generated 98 suggestions and 1,329 votes from 228 users in March and April 2012.

Progress on ongoing updates to the technical standards for Section 508 has been steadier than compliance and enforcement. Originally issued in December 2000, the 508 technical guidelines have not kept up with advancements in Web technology or with complementary W3C standards. A first draft of revised Section 508 standards was issued in March 2010. A revision issued in December 2011 marked a change of course, suggesting use of the W3C's WCAG 2.0 proposals as the benchmark for accessibility of federal websites; the Section 508 refresh also included updates to Section 255 of the Telecommunications Act of 1996.[10] It is expected the U.S. Access Board will release a third, more final draft by the end of 2013, with implementation following a period of public comment.

Americans with Disabilities Act (1990). The three sections (titles) of the ADA cover employment of people with disabilities, state and local government services, and accessibility of public accommodations. Although the original version of the ADA did not

For More Information

Section 504 of the Rehabilitation Act; http://www2.ed.gov/about/offices/list/ocr/504faq.html

Americans with Disabilities Act; http://www.ada.gov

Section 508 specifications; http://www.section508.gov

W3C Web Accessibility Initiative; http://www.w3.org/WAI/

Web Content Accessibility Guidelines; http://www.w3.org/TR/WCAG20/

W3C Accessible Rich Internet Applications recommendation; http://www.w3.org/TR/wai-aria-practices/

Individuals with Disabilities Education Act (for K–12 education); http://idea.ed.gov/

Help America Vote Act; http://www.eac.gov/about_the_eac/help_america_vote_act.aspx

White House invitation for suggestions for improving Section 508 implementation; http://section508.ideascale.com/

21st Century Communications and Video Accessibility Act; https://www.fcc.gov/guides/21st-century-communications-and-video-accessibility-act-2010

IT accessibility laws outside of U.S through ACM interactions magazine's forum on interacting with public policy; http://www.sigchi.org/about/sigchi-public-policy

W3C Web Accessibility Initiative on government policy; http://www.w3.org/WAI/Policy/

ACM U.S. Public Policy Council; http://usacm.acm.org

ACM SIGs, including ASSETS http://www.sigaccess.org/ and CHI http://www.sigchi.org

U.S. disability rights law (blog); http://disabilitylaw.blogspot.com

specifically mention certain topics (such as e-books and websites), court rulings and Department of Justice statements indicate information technology is covered under the ADA. Title III delineates 12 categories of public accommodations that must be accessible, including stores, libraries, travel facilities, and recreational facilities.

Although the Department of Justice has publicly said since 1996 that the ADA applied to websites of public accommodations, the legal precedent was not established until a preliminary ruling in *National Federation of the Blind vs. Target* (2006–2008) found that websites are services of places of public accommodation and therefore covered by the ADA. Websites of public accommodations fall under Title III, except for state and local government websites, which fall under Title II.

In a 2012 case (*National Association of the Deaf vs. Netflix*), the U.S. District Court for the District of Massachusetts ruled that services provided through the customer's home, even without a corresponding physical place of public accommodation, were also covered by the ADA if they provide the services of a place of public accommodation (such as a rental establishment and place of exhibition or entertainment). This ruling effectively means the ADA applies to Web-only businesses when they provide goods or services that would normally be covered under the ADA if they took place in a physical establishment. In an October 2012 consent decree, Netflix agreed to provide captioning on 100% of its online content by 2014.

In July 2010, the Department of Justice began the process of rulemaking to implement specific guidelines for website ADA compliance. Until there are specific regulations and technical guidelines for the ADA, compliance can be met by following either the WCAG international standards or the Section 508 technical standards, although there is no legal or regulatory guidance on how to evaluate for accessibility or how often it must be monitored. The comment period on the July 2010 notice continued until January 2011, and a first draft of the guidelines for websites of public accommodations could be released by the end of 2013. Earlier in 2013, the rulemaking process for website accessibility under Title II (state and local government) was separated from Title III (public accommodations); the rulemaking for state and local governments, as a separate process, is expected to be completed more quickly than the regulatory process for public accommodations. Unclear is how regulations due to Title II will affect similar state laws.

Accessibility of information technology in higher education continues to be a focus of federal enforcement efforts. Because all universities (public and private) receive federal funding, all are covered under both the ADA and Section 504 of the Rehabilitation Act. A 2010 letter from the Departments of Justice and Education to all university presidents advising of the legal requirements related to accessible instructional materials[12] was followed by multiple settlements with universities using inaccessible e-book readers, course-management systems, and library websites. A congressional commission issued a report on accessible instructional materials in December 2011,[13] followed by Senate hearings in February 2012. New legislative proposals and/or guidelines regarding accessibility of IT in higher education are expected to be introduced over the next few years.

Signed October 8, 2010 by President Barack Obama, the 21st Century Communications and Video Accessibility Act (CVAA) modernized the Communications Act of 1934 to apply to modern communications devices. Specifically, it requires accessibility features on new smartphones, captioning of video content delivered through Internet protocol, accessibility of emergency-warning information distributed in any format, accessible menus on DVDs and televisions, and accessible emergency services. It also calls for creation of a Web-based clearinghouse for information on accessible communication devices. The accessibility clearinghouse was established by the Federal Communications Commission (FCC) in October 2011.

The CVAA requires a biennial report to Congress on progress implementing accessibility requirements. Issued by the FCC October 5, 2012, the first report detailed extensive initial efforts, including notices of proposed rulemaking seeking comment on CVAA-related issues, five reports, and convening of advisory committees.[3] The FCC's assessment of progress on meeting accessibility goals was mixed; although some progress was evident, accessibility advocates generally felt much more was needed.

As the newest federal law on accessibility of information technology, the 2010 CVAA provides an informative case study of the opportunities and challenges facing computing professionals interested in influencing policy discussions. Closed-captioning requirements for video transmitted through Internet protocol demonstrate the concerns associated with codification of technical and functional requirements. The CVAA requires closed-captioning functionality on devices designed for video playback, with exceptions for smaller devices (screen size smaller than 13 inches) required to provide such features only if "achievable." Although perhaps intended to minimize technical challenges for developers of smaller devices, codification of specific display sizes could discourage innovation and delay adoption of new devices by users concerned about potential lack of accessibility. However, the clause is already obsolete. As many smartphone and tablet users know, devices significantly smaller than 13 inches are quite suitable for use with closed captions. Requirements omitting specific mention of screen size would arguably encourage greater accessibility without creating unrealistic expectations.

Discussion

The complex legal landscape of overlapping laws and regulations plays a significant role in fueling the perception of accessibility as "hard." Uncertainty as to which accessibility requirements must be met under which circumstances leaves developers in a quandary: When are their products, software, and websites accessible "enough"? To promote accessibility while minimizing undue burden, laws and regulations must provide clear guidance on policies, indicating which organizations are covered and how accessibility is technically and legally ascertained or measured.

Effective accessibility policies must focus on functional, not technical requirements. Rather than issue potentially obsolescent rules based on specific solutions, effective policies spell out goals that encourage designers to innovate in support of accessibility. Regulations that specify specific solutions to accessibility problems (such as "text-only" websites discussed earlier) or codify technical parameters (such as the 13-inch screen size specified in the CVAA) may discourage innovative design (such as the Accessible Rich Internet Applications proposals). More effective policies would instead focus on the requirement that information technologies provide users with no- or low-vision resources that are functionally equivalent to those available to visual users. After all, who in the past would have expected touchscreens to ever be recognized as an effective way to meet the accessibility needs of blind users, as is currently the case? Functional requirements would have allowed for this innovative approach, while technical requirements likely would not have included touchscreens as an appropriate accommodation.

Proposed policies should also be closely examined by all stakeholders for potential unintended adverse consequences. Overly stringent accessibility requirements could, for example, leave providers of legacy documents with the mistaken perception that they must choose between incurring the potentially large expense of updating all legacy documents for accessibility or removing content to be nominally in compliance with regulations. However, such misunderstandings are often due to a lack of understanding of accessibility laws and regulations that often recognize the need for gradual change. When the ADA was introduced almost 25 years ago, the physical accessibility requirements immediately applied to new construction or renovated buildings, while less demanding standards were specified for older existing buildings.[1] Many policies related to accessibility of information technology recognize the need for gradual change; for example, Oregon State University and the University of Wisconsin (in meeting their ADA and 504 requirements) have implemented similar policies for improving accessibility of Web content. Although all newly posted content must be accessible by following accessibility guidelines, the updating of previously existing "legacy" content for accessibility is prioritized based on use, with accessibility efforts starting with most frequently used pages, as well as with any pages related to disability services. Infrequently used pages are made accessible upon request. In the ongoing rulemaking process to develop clearer guidelines within the ADA for Web content of public accommodations (discussed earlier), the Department of Justice specifically asked the community for feedback on how to deal with legacy content.

With essential government services increasingly moving to the Web, failure to support accessibility means denial of service to those who need information the most.

Given these complexities, some observers might suggest economic concerns alone could succeed in encouraging accessibility. What business could possibly afford to alienate large blocks of potential customers, particularly if its competitors' sites and products are more accessible? The combination of increased access to consumers and protection from legal liability associated with inaccessible sites and products should theoretically provide economic incentives to move toward improved accessibility, especially when accessibility is not a major cost. However, achieving accessibility is not as simple as "let the market forces take care of it."

Economic concerns are not driving motivators for government agencies and other not-for-profit entities that do not measure success solely in terms of sales numbers. With essential government services increasingly moving to the Web, failure to support accessibility means denial of service to those who need information the most. For commercial websites, loss of sales has been shown to be, for some, insufficient motivation. A long history of companies foregoing profits, rather than servicing customers with disabilities, demonstrates that decisions relating to accessibility are not always made in an economically rational way.[9]

Although some IT providers fear the costs of accessibility, such concerns are likely overstated. Effective integration of accessibility will account for accessibility concerns throughout the development life cycle, at minimal cost. As with physical architecture, where the cost of accessibility in new designs is small compared to the cost of retrofitting after the fact, incorporating accessibility features early in design may add only 2% to 3% to the cost and time of development.[14] Products and services accessible from the outset provide all users immediate access, while leaving accessibility features for "version 2.0" effectively discriminates against users who need assistive features today.

Embracing accessibility might also help technology providers avoid exposure to potential liability under laws that do not necessarily focus on accessibility. Websites that provide special online-only pricing for goods or services might be engaging in illegal pricing discrimination if their designs are either inaccessible[6] or do not consistently provide accommodations required by law.[4] The resulting violations could involve multiple laws, some possibly risking more serious penalties than those associated with the disability rights laws described here. Likewise, recruiting websites or employers requiring online applications without providing equal access for people with disabilities could be engaging in illegal forms of employment discrimination.[5]

Accessibility is an issue outside the U.S. as well. Just as in the U.S., laws in other countries typically start by requiring accessible government information. Australia, Canada, France, Spain, and others have laws relating to the accessibility of government information technology. The U.K., with its Equality Act of 2010, was one of the first countries outside the U.S. to require accessibility in the greater community for companies and organizations. For additional discussion of IT accessibility laws outside of the U.S., see *ACM interactions* magazine's forum on interacting with public policy and the W3C's Web Accessibility Initiative on government policy.

Get Involved

Weighing the trade-offs in policy requires consideration of costs and benefits from the perspectives of multiple interested parties, including individuals in need of accessible technology, educational institutions, and businesses and organizations that provide information and services. Due to the important effect that laws and regulations have on interaction designers, software engineers, Web developers, and other IT professionals, it is important to not only understand how the laws affect developers but also to be involved in the policy-making process.

Computing professionals interested in promoting accessibility can engage with lawmakers, advocacy groups, and industry partners to share their understanding of accessibility and encourage development of appropriate technology and policy. Comments on proposed legislation and regulation could raise concern as to definitions or implementation challenges, identify potential unintended consequences, suggest alternative approaches that might encourage accessibility, and educate legislators and rulemakers who might not be well versed in the technical details. Participation in bodies like the W3C can help develop widely used approaches for addressing accessibility concerns.

ACM has been involved since 1992 through its ACM U.S. Public Policy Council (USACM) and accessibility subcommittee. USACM actions on accessibility include a statement advocating Web-accessibility principles and comments on various policy proposals related to the ADA, Section 508, and accessibility of electronic health records. Membership in USACM is open to all interested ACM members in the U.S. ACM seeks to educate policymakers and the public, as well as the computer science community, on science and technology. ACM engages through USACM for technology policy and the Education Policy Committee for educational matters. ACM special interest groups, including SIGACCESS, SIGCHI, and SIGWEB, often engage in accessibility issues; conferences, including ASSETS and CHI, showcase accessibility research.

Computing professionals have much to contribute to the discussion. By suggesting opportunities for better design, outlining paths to affordable accessibility, and constructively critiquing policy proposals aiming to avoid undesired (and unintended) consequences, ACM members can help make accessibility much less hard.

For more on U.S. disability rights law, see the blog by Sam Bagenstos, a professor at the University of Michigan School of Law and former Deputy Assistant Attorney General in the Department of Justice. C

References

1. Bagenstos, S. *Disability Rights Law: Cases and Materials.* Thompson Reuters, New York, 2010.
2. Cerf, V. Why is accessibility so hard? *Commun. ACM 55*, 11 (Nov. 2012). 7.
3. Federal Communications Commission. *Biennial Report to Congress as Required by the 21st Century Communications and Video Accessibility Act of 2010.* Washington, D.C., 2012; http://www.fcc.gov/document/cvaa-report-congress
4. Lazar, J., Jaeger, P., Adams, A. et al. Up in the air: Are airlines following the new DOT rules on equal pricing for people with disabilities when websites are inaccessible? *Government Information Quarterly 27*, 4 (Oct. 2010), 329–336.
5. Lazar, J., Olalere, A., and Wentz, B. Investigating the accessibility and usability of job application websites for blind users. *Journal of Usability Studies 7*, 2 (Feb. 2012), 68–87.
6. Lazar, J., Wentz, B., Bogdan, M. et al. Potential pricing discrimination due to inaccessible web sites. In *Proceedings of INTERACT* (Lisbon, Portugal, Sept. 5–9). Springer, London, 2011, 108–114.
7. Office of Management and Budget. *Improving the Accessibility of Government Information.* Washington, D.C., 2010; http://www.whitehouse.gov/sites/default/files/omb/assets/procurement_memo/improving_accessibility_gov_info_07192010.pdf
8. Shapiro, J. *No Pity: People with Disabilities Forging a New Civil Rights Movement.* Random House, New York, 1994.
9. Stein, M. The law and economics of disability accommodations. *Duke Law Journal 53*, 1 (Oct. 2003), 79–191.
10. U.S. Access Board. *Draft Information and Communication Technology Standards and Guidelines.* Washington, D.C., 2011; http://www.access-board.gov/sec508/refresh/draft-rule.htm
11. U.S. Department of Justice. *Section 508 Report to the President and Congress: Accessibility of Federal Electronic and Information Technology.* Washington, D.C., 2012; http://www.ada.gov/508/508_Report.htm
12. U.S. Department of Education. *Joint 'Dear Colleague' Letter: Electronic Book Readers.* Washington, D.C., 2010; http://www2.ed.gov/about/offices/list/ocr/letters/colleague-20100629.html
13. U.S. Department of Education. *Report of the Advisory Commission on Accessible Instructional Materials in Postsecondary Education for Students with Disabilities.* Washington, D.C., 2011; http://www2.ed.gov/about/bdscomm/list/aim/publications.html
14. Wentz, B., Jaeger, P., and Lazar, J. Retrofitting accessibility: The inequality of after-the-fact access for persons with disabilities in the United States. *First Monday 16*, 11 (Nov. 2011) http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/3666/3077

Jonathan Lazar (jlazar@towson.edu) is a professor of computer and information sciences at Towson University, Towson, MD; this work was completed while he was the Shutzer Fellow in the Radcliffe Institute for Advanced Study at Harvard University, Cambridge, MA. He is also chair of public policy for ACM SIGCHI and director of the Universal Usability Laboratory at Towson University.

Harry Hochheiser (harryh@pitt.edu) is an assistant professor of biomedical informatics at the University of Pittsburgh School of Medicine and chair of the accessibility subcommittee of the USACM, the ACM U.S. Public Policy Committee.

Be more than a word on paper.

UNI FRESHMAN
QUALS MENTORS MAJOR
CUM LAUDE EXAMS PROFESSORS
HIGH SCHOOL GRADUATION TUMBLR
COMP SCI THESIS TECHNOLOGY GLOBAL
SEMESTER XRDS QUALITY JUNIOR
VOLUNTEER EDUCATION COLLEGE
SENIOR FINALS TECH
VOICE EDITORS CHANGE
COMMUNITY CREATIVITY CS
CHALLENGE TEAM WORK LAB ACM
LECTURE RECOGNITION SOPHOMORE
DINING HALL
FACEBOOK CONTROL GRAD SCHOOLS
PRECEPT PHD SCHOLARSHIP
ACADEMIA DISSERTATION DEFENSE
LIBRARY ADVISOR TWITTER
SPRING BREAK

Join *XRDS* as a student editor and
be the voice for students worldwide.

If you are interested in volunteering as a
student editor, please contact xrds@hq.acm.org with
"Student Editor" in the subject line.

XRDS

acm Association for Computing Machinery

DOI:10.1145/2500500

A broader class of consistency guarantees can, and perhaps should, be offered to clients that read shared data.

BY DOUG TERRY

Replicated Data Consistency Explained Through Baseball

REPLICATED STORAGE SYSTEMS for the cloud deliver different consistency guarantees to applications that are reading data. Invariably, cloud storage providers redundantly store data on multiple machines so that data remains available in the face of unavoidable failures. Replicating data across datacenters is not uncommon, allowing the data to survive complete site outages. However, the replicas are not always kept perfectly synchronized. Thus, clients that read the same data object from different servers can potentially receive different versions.

Some systems, like Microsoft's Windows Azure, provide only strongly consistent storage services to their applications.[5] These services ensure clients of Windows Azure Storage always see the latest value that was written for a data object. While strong consistency is desirable and reasonable to provide within a datacenter, it raises concerns as systems start to offer geo-replicated services that span multiple datacenters on multiple continents.

Many cloud storage systems, such as the Amazon Simple Storage Service (S3), were designed with weak consistency based on the belief that strong consistency is too expensive in large systems. The designers chose to relax consistency in order to obtain better performance and availability. In such systems, clients may perform read operations that return stale data. The data returned by a read operation is the value of the object at *some past point in time* but not necessarily the latest value. This occurs, for instance, when the read operation is directed to a replica that has not yet received all of the writes that were accepted by some other replica. Such systems are said to be *eventually consistent*.[12]

Recent systems, recognizing the need to support different classes of applications, have been designed with a choice of operations for accessing cloud storage. Amazon's DynamoDB, for example, provides both *eventually consistent reads* and *strongly consistent reads*, with the latter experiencing a higher read latency and a twofold reduction in read throughput.[1] Amazon SimpleDB offers the same choices for clients that

» key insights

- **Although replicated cloud services generally offer strong or eventual consistency, intermediate consistency guarantees may better meet an application's needs.**
- **Consistency guarantees can be defined in an implementation-independent manner and chosen for each read operation.**
- **Dealing with relaxed consistency need not place an excessive burden on application developers.**

PHOTOGRAPH FROM SHUTTERSTOCK.COM

read data. Similarly, the Google App Engine Datastore added eventually consistent reads to complement its default strong consistency.[8] PNUTS, which underlies many of Yahoo's Web services, provides three types of read operations: *read-any*, *read-critical*, and *read-latest*.[7] Modern quorum-based storage systems allow clients to choose between strong and eventual consistency by selecting different read and write quorums.[4]

In the research community over the past 30 years, a number of consistency models have been proposed for distributed and replicated systems.[10] These offer consistency guarantees that lie somewhere in between strong consistency and eventual consistency. For example, a system might guarantee that a client sees data that is no more than five minutes out of date or that a client always observes the results of its own writes. Actually, some consistency models are even weaker than eventual consistency, but those I ignore as being less than useful.

The reason for exploring different consistency models is that there are fundamental trade-offs between consistency, performance, and availability.[9,10,12,13] Offering stronger consistency generally results in lower performance and reduced availability for reads or writes or both. The CAP theorem has proven that, for systems

Table 1. Six consistency guarantees.

Strong Consistency	See all previous writes.
Eventual Consistency	See subset of previous writes.
Consistent Prefix	See initial sequence of writes.
Bounded Staleness	See all "old" writes.
Monotonic Reads	See increasing subset of writes.
Read My Writes	See all writes performed by reader.

that must tolerate network partitions, designers must choose between consistency and availability.[5] In practice, latency is an equally important consideration.[1] Each proposed consistency model occupies some point in the complex space of trade-offs.

Are different consistencies useful in practice? Can application developers cope with eventual consistency? Should cloud storage systems offer an even greater choice of consistency than the consistent and eventually consistent reads offered by some of today's services?

This article attempts to answer these questions, at least partially, by examining an example (but clearly fictitious) application: the game of baseball. In particular, I explore the needs of different people who access the score of a baseball game, including the scorekeeper, umpire, radio reporter, sportswriter, and statistician. Supposing the score is stored in a cloud-based, replicated storage service, I show eventual consistency is insufficient for most of the participants, but strong consistency is not needed either. Most participants benefit from some intermediate consistency guarantee.

The next section defines six possible consistency guarantees for read operations. Then I present an algorithm that emulates a baseball game, indicating where data is written and read, and I enumerate the results that might be returned when reading the score with different guarantees. I also examine the roles of various people who want to access the baseball score and the read consistency that each desires and draw conclusions from this simple example.

Read Consistency Guarantees

While replicated systems have provided many types of data consistency over the past 30 years, and a wide variety of consistency models have been explored in the computer science research community, many of these are tied to specific implementations. Frequently, one needs to understand how a system operates in order to understand what consistency it provides in what situations. This places an unfortunate burden on those who develop applications on top of such storage systems.

The six consistency guarantees I advocate here can be described in a simple, implementation-independent way. This not only benefits application developers but also can permit flexibility in the design, operation, and evolution of the underlying storage system.

These consistency guarantees are based on a simple model in which clients perform *read* and *write* operations to a data store. Multiple clients may concurrently access shared information, such as social network graphs, news feeds, photos, shopping carts, or financial records. The data is replicated among a set of servers, but the details of the replication protocol are hidden from clients. A write is any operation that updates one or more data objects. Writes are eventually received at all servers and performed in the same order. This order is consistent with the order in which clients submit write operations. In practice, the order could be enforced by performing all writes at a master server or by having servers run a consensus protocol to reach agreement on the global order. Reads return the values of one or more data objects that were previously written, though not necessarily the latest values. Each read operation can request a consistency guarantee, which dictates the set of allowable return values. Each guarantee is defined by the set of previous writes whose results are visible to a read operation. Table 1 summarizes these six consistency guarantees.

Strong consistency is particularly easy to understand. It guarantees a read operation returns the value that was last written for a given object. If write operations can modify or extend portions of a data object, such as appending data to a log, then the read returns the result of applying all writes to that object. In other words, a read observes the effects of *all* previously completed writes.

Eventual consistency is the weakest of the guarantees, meaning it allows the greatest set of possible return values. For whole-object writes, an eventually consistent read can return any value for a data object that was written in the past. More generally, such a read can return results from a replica that has received an arbitrary subset of the writes to the data object being read. The term "eventual" consistency derives from the fact that each replica eventually receives each write operation, and if clients stopped performing writes then read operations would eventually return an object's latest value.

By requesting a *consistent prefix*, a reader is guaranteed to observe an ordered sequence of writes starting with the first write to a data store. For example, the read may be answered by a replica that receives writes in order from a master replica but has not yet received some recent writes. In other words, the reader sees a version of the data store that existed at the master at some time in the past. This is similar to the "snapshot isolation" consistency offered by many database management systems. For reads to a single data object in a system where write operations completely overwrite previous values of an object, even eventual consistency reads observe a consistent prefix. The main benefit of requesting a consistent prefix arises when reading multiple data objects or when write operations incrementally update an object.

Bounded staleness ensures read results are not too out of date. Typically, staleness is defined by a time period T, say five minutes. The storage system guarantees a read operation will return any values written more than T minutes ago or more recently written values. Alternative, some systems have defined staleness in terms of the number of missing writes or even the amount of inaccuracy in a data value. I find that time-bounded staleness is the most natural concept for application developers.

Monotonic reads is a property that applies to a sequence of read operations performed by a given storage system client. As such, it is called a "session guarantee."[11] With monotonic reads, a client can read arbitrarily stale data, as with eventual consistency, but is guaranteed to observe a data store that is increasingly up to date over time. In particular, if the client issues a read operation and then later issues another read to the same object(s), the second read will return the same value(s) or a more recently written value.

Read my writes is a property that also applies to a sequence of operations performed by a single client. It guarantees the effects of all writes that were performed by the client are visible to the client's subsequent reads. If a client writes a new value for a data object and then reads this object, the read will return the value that was last written by the client (or some other value that was later written by a different client). For clients that have issued no writes, the guarantee is the same as eventual consistency. (Note: In previous articles this has been called "Read Your Writes,"[11] but I have chosen to rename it to more accurately describe the guarantee from the client's viewpoint.)

These last four read guarantees are all a form of eventual consistency but stronger than the eventual consistency model that is typically provided in cloud storage systems. The "strength" of a consistency guarantee does not depend on when and how writes propagate between servers, but rather is defined by the size of the set of allowable results for a read operation. Smaller sets of possible read results indicate stronger consistency. When requesting strong consistency, there is a single value that must be returned, the latest value that was written. For an object that has been updated many times, an eventually consistent read can return one of many suitable values. Of the four intermediate guarantees, none is stronger than any of the others, meaning each might have a different set of possible responses to a read operation. In some cases, as will be shown later, applications may want to request multiple guarantees. For example, a client could request both monotonic reads and read my writes so that it observes a data store that is consistent with its own actions.[11]

In this article, the data store used for baseball scores is a traditional key-value store, popularized by the "noSQL" movement. Writes, also called *puts*, modify the value associated with a given key. Reads, also called *gets*, return the value for a key. However, these guarantees can apply to other types of replicated data stores with other types of read and write operations, such as file systems and relational databases. This is why the guarantees are defined in terms of writes rather than data values. For example, in a system that offers an increment or an append operation, all writes performed on an object contribute to the object's observed value, not just the latest write. Moreover, the guarantees could apply to atomic transactions that access multiple objects, though the examples in this article do not require atomic updates.

Table 2 shows the performance and availability typically associated with each consistency guarantee. It rates the three properties on a scale from poor to excellent. Consistency ratings are based on the strength of the consistency guarantee as previously defined. Performance refers to the time it takes to complete a read operation, that is, the read latency. Availability is the likelihood of a read operation successfully returning suitably consistent data in the presence of server failures.

Strong consistency is desirable from a consistency viewpoint but offers the worst performance and availability since it generally requires reading from a designated primary site or from a majority of replicas. Eventual consistency, on the other hand, allows clients to read from any replica, but offers the weakest consistency. The inverse correlation between performance and consistency is not surprising since weaker forms of consistency generally permit read requests to be sent to a wider set of servers. With more choices of servers that are sufficiently up to date, clients are more able to choose a nearby server. The latency difference between accessing a local rather than a remote server can be a factor of 100. Similarly, a larger choice of servers means a client is more likely to find one (or a quorum) that is reachable, resulting in higher availability.

Each guarantee offers a unique combination of consistency, performance, and availability. Labeling each cell in Table 2 is not an exact science (and I could devote a whole article to this topic). One might argue that some entry listed as "okay" should really be "good", or vice versa, and indeed the characteristics do depend to some extent on implementation, deployment, and operating details. For some clients, eventually consistent reads may often return strongly consistent results, and may not be any more efficient than strongly consistent reads.[3,13] But, the general comparisons between the various consistency guarantees are qualitatively accurate. The bottom line is that one faces substantial trade-offs when choosing a particular replication scheme with a particular consistency model.

Without offering any evidence, I assert that all of these guarantees can be provided as choices within

Table 2. Consistency, performance, and valuability trade-offs.

Guarantee	Consistency	Performance	Availability
Strong Consistency	excellent	poor	poor
Eventual Consistency	poor	excellent	excellent
Consistent Prefix	okay	good	excellent
Bounded Staleness	good	okay	poor
Monotonic Reads	okay	good	good
Read My Writes	okay	okay	okay

Figure 1. A simplified baseball game.

```
Write ("visitors", 0);
Write ("home", 0);
for inning = 1 .. 9
   outs = 0;
   while outs < 3
     visiting player bats;
     for each run scored
        score = Read ("visitors");
        Write ("visitors", score + 1);
   outs = 0;
   while outs < 3
      home player bats;
      for each run scored
         score = Read ("home");
         Write ("home", score + 1);
end game;
```

the same storage system. In fact, my colleagues and I at the MSR Silicon Valley Lab have built a prototype of such a system (but that is the topic for another article). In our system, clients requesting different consistency guarantees experience different performance and availability for the read operations they perform, even when accessing shared data. Here, let's assume the existence of a storage system that offers its clients a choice of these six read guarantees. I proceed to show how they would be used…in baseball.

Baseball as a Sample Application

For those readers who are not familiar with baseball, but who love to read code, Figure 1 illustrates the basics of a nine-inning baseball game. The game starts with the score of 0-0. The visitors bat first and remain at bat until they make three outs. Then the home team bats until it makes three outs. This continues for nine innings. Granted, this leaves out many of the subtleties that are dear to baseball aficionados, like myself. But it does explain all that is needed for this article.

Assume the score of the game is recorded in a key-value store in two objects, one for the number of runs scored by the "visitors" and one for the "home" team's runs. When a team scores a run, a read operation is performed on its current score, the returned value is incremented by one, and the new value is written back to the key-value store.

Figure 2. Sequence of writes for a sample game.

```
Write ("home", 1)
Write ("visitors", 1)
Write ("home", 2)
Write ("home", 3)
Write ("visitors", 2)
Write ("home", 4)
Write ("home", 5)
```

As a concrete example, consider the write log for a sample game as shown in Figure 2. In this game, the home team scored first, then the visitors tied the game, then the home team scored twice more, and so on.

This sequence of writes could be from a baseball game with the inning-by-inning line score that is illustrated in Figure 3. This hypothetical game is currently in the middle of the seventh inning (the proverbial seventh-inning stretch), and the home team is winning 2-5.

Suppose the key-value store that holds the visitors and home team's run totals resides in the cloud and is replicated among a number of servers. Different read guarantees may result in clients reading different scores for this game that is in progress. Table 3 lists the complete set of scores that could be returned by reading the visitors and home scores with each of the six consistency guarantees. Note that the visitors' score is listed first, and different possible return values are separated by comas.

A strong consistency read can only return one result, the current score, whereas an eventual consistency read can return one of 18 possible scores. Observe that many of the scores that can be returned by a pair of eventually consistent reads are ones that were never the actual score. For example, reading the visitors' score may return two and reading the home team's score may return zero, even though the home team never trailed. The consistent prefix property limits the result to scores that actually existed at some time. The results that can be returned by a bounded staleness read clearly depend on the desired bound. Table 3 illustrates the possible scores for a bound of one inning, that is, scores that are at most one inning out of date; for a bound of seven innings or more,

Figure 3. The line score for this sample game.

	1	2	3	4	5	6	7	8	9	RUNS
Visitors	0	0	1	0	1	0	0			2
Home	1	0	1	1	0	2				5

Table 3. Possible scores read for each consistency guarantee.

Strong Consistency	2-5
Eventual Consistency	0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5
Consistent Prefix	0-0, 0-1, 1-1, 1-2, 1-3, 2-3, 2-4, 2-5
Bounded Staleness	scores that are at most one inning out-of-date: 2-3, 2-4, 2-5
Monotonic Reads	after reading 1-3: 1-3, 1-4, 1-5, 2-3, 2-4, 2-5
Read My Writes	for the writer: 2-5 for anyone other than the writer: 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5

Figure 4. Role of the scorekeeper.

```
score = Read ("visitors");
Write ("visitors", score + 1);
```

Figure 5. Role of the umpire.

```
if first half of 9th inning complete then
    vScore = Read ("visitors");
    hScore = Read ("home");
    if vScore < hScore
        end game;
```

the result set is the same as for eventual consistency in this example. In practice, a system is unlikely to express staleness bounds in units of "innings." So, for this example, assume the reader requested a bound of 15 minutes and the previous inning lasted exactly that long. For monotonic reads, the possible return values depend on what has been read in the past. For read my writes they depend on who is writing to the key-value store; in this example, assume all of the writes were performed by a single client.

Read Requirements for Participants

Now, let's examine the consistency needs of a variety of people involved in a baseball game who want to read the score. Certainly, each of these folks could perform a strongly consistent read to retrieve the visiting and home team's score. In this case, as pointed out in the previous section, only one possible value would be returned: the current score. However, as shown in Table 2, readers requesting strong consistency will likely receive longer response times and may even find that the data they are requesting is not currently available due to temporary server failures or network outages. The point of this section is to evaluate, for each participant, the minimum consistency that is required. By requesting read guarantees that are weaker than strong consistency, these clients are likely to experience performance benefits and higher availability.

Official scorekeeper. The official scorekeeper is responsible for maintaining the score of the game by writing it to the persistent key-value store. Figure 4 illustrates the steps taken by the scorekeeper each time the visiting team scores a run; his action when the home team scores is similar. Note that this code is a snippet of the overall baseball game code that was presented in in Figure 1.

What consistency does the scorekeeper require for his read operations? Undoubtedly, the scorekeeper needs to read the most up-to-date previous score before adding one to produce the new score. Otherwise, the scorekeeper runs the risk of writing an incorrect score and undermining the game, not to mention inciting a mob of angry baseball fans. Suppose the home team had previously scored five runs and just scored the sixth. Doing an eventual consistency read, as shown in Table 3, could return a score of anything from zero to five. Perhaps, the scorekeeper would get lucky and receive the correct score in response to his read, but he should not count on it.

Interestingly, while the scorekeeper requires strongly consistent data, he does not need to perform strong consistency reads. Since the scorekeeper is the only person who updates the score, he can request the read my writes guarantee and receive the same effect as a strong read. Essentially, the scorekeeper uses application-specific knowledge to obtain the benefits of a weaker consistency read without actually giving up any consistency.

This might seem like a subtle distinction, but, in fact, could be quite significant in practice. In processing a strong consistency read the storage system must pessimistically assume that some client, anywhere in the world, may have just updated the data. The system therefore must access a majority of servers (or a fixed set of servers) in order to ensure the most recently written data is accessed by the submitted read operation. In providing the read my writes guarantee, on the other hand, the system simply needs to record the set of writes that were previously performed by the client and find some server that has seen all of these writes.[11] In a baseball game, the previous run that was scored, and hence the previous write that was performed by the scorekeeper, may have happened many minutes or even hours ago. In this case, almost any server will have received the previous write and be able to answer the next read that requests the read my writes guarantee.

Figure 6. Role of the radio sports reporter.

```
do {
    vScore = Read ("visitors");
    hScore = Read ("home");
    report vScore and hScore;
    sleep (30 minutes);
}
```

Umpire. The umpire is the person who officiates a baseball game from behind home plate. The umpire, for the most part, does not actually care about the current score of the game. The one exception comes after the top half of the 9th inning, that is, after the visiting team has batted and the home team is about to bat. Since this is the last inning (and a team cannot score negative runs), the home team has already won if they are ahead in the score; thus, the home team can and does skip its last at bat in some games. The code for the umpire who needs to make this determination is illustrated in Figure 5.

When accessing the score during the 9th inning, the umpire does need to read the current score. Otherwise, he might end the game early, if he incorrectly believes the home team to be ahead, or make the home team bat unnecessarily. Unlike the scorekeeper, the umpire never writes the score; he simply reads the values that were written by the official scorekeeper. Thus, in order to receive up-to-date information, the umpire must perform strong consistency reads.

Radio reporter. In most areas of the U.S., radio stations periodically announce the scores of games that are in progress or have completed. In the San Francisco area, for example, KCBS reports sports news every 30 minutes. The radio reporter performs the steps outlined in Figure 6. A similar, perhaps more modern, example is the sports scores that scroll across the bottom of the TV screen while viewers are watching ESPN.

Figure 7. Role of the sportswriter.

```
While not end of game {
    drink beer;
    smoke cigar;
}
go out to dinner;
vScore = Read ("visitors");
hScore = Read ("home");
write article;
```

Figure 8. Role of the statistician.

```
Wait for end of game;
score = Read ("home");
stat = Read ("season-runs");
Write ("season-runs", stat + score);
```

Figure 9. Role of the stat watcher.

```
do {
    stat = Read ("season-runs");
    discuss stats with friends;
    sleep (1 day);
}
```

If the radio reporter broadcasts scores that are not completely up to date, that is okay. People are accustomed to receiving old news. Thus, some form of eventual consistency is fine for the reads he performs. But what guarantees, if any, are desirable?

As shown in Table 3, the read with the weakest guarantee, an eventual consistency read, may return scores that never existed. For the sample line score given in Figure 3, such a read might return a score with the visitors leading 1-0, even though the visiting team has never actually been in the lead. The radio reporter does not want to report such fictitious scores. Thus, the reporter wants both his reads to be performed on a snapshot that hold a consistent prefix of the writes that were performed by the scorekeeper. This allows the reporter to read the score that existed at some time, without necessarily reading the current score.

But reading a consistent prefix is not sufficient. For the line score in Figure 3, the reporter could read a score of 2-5, the current score, and then, 30 minutes later, read a score of 1-3. This might happen, for instance, if the reporter happens to read from a primary server and later reads from another server, perhaps in a remote datacenter, that has been disconnected from the primary and has yet to receive the latest writes. Since everyone knows that baseball scores are monotonically increasing, reporting scores of 2-5 and 1-3 in subsequent news reports would make the reporter look foolish. This can be avoided if the reporter requests the monotonic reads guarantee in addition to requesting a consistent prefix. Observe that neither guarantee is sufficient by itself.

Alternatively, the reporter could obtain the same effect as a monotonic read by requesting bounded staleness with a bound of less than 30 minutes. This would ensure the reporter observes scores that are at most 30 minutes out of date. Since the reporter only reads data every 30 minutes, he must receive scores that are increasingly up to date. Of course, the reporter could ask for a tighter bound, say five minutes, to get scores that are reasonably timely.

Sportswriter. Another interesting person is the sportswriter who watches the game and later writes an article that appears in the morning paper or that is posted on some website. Different sportswriters may behave differently, but my observations (from having been a sportswriter) is they often act as in Figure 7.

The sportswriter may be in no hurry to write his article. In this example, he goes out to a leisurely dinner before sitting down to summarize the game. He certainly wants to make sure that he reports the correct final score for the game. So, he wants the effect of a strong consistency read. However, he does not need to pay the cost. If the sportswriter knows he spent an hour eating dinner after the game ended, then he also knows it has been at least an hour since the scorekeeper last updated the score. Thus, a bounded staleness read with a bound of one hour is sufficient to ensure the sportswriter reads the final score. In practice, any server should be able to answer such a read. In fact, an eventual consistency read is likely to return the correct score after an hour, but requesting bounded staleness is the only way for the sportswriter to be 100% certain he is obtaining the final score.

Statistician. The team statistician is responsible for keeping track of the season-long statistics for the team and for individual players. For example, the statistician might tally the total number of runs scored by her team this season. Suppose these statistics are also saved in the persistent key-value store. As shown in Figure 8, the home team's statistician, sometime after each game has ended, adds the runs scored to the previous season total and writes this new value back into the data store.

When reading the team's score from today, the statistician wants to be sure to obtain the final score. Thus, she needs to perform a strong consistency read. If the statistician waits for some time after the game, then a bounded staleness read may achieve the same effect (as discussed earlier for the sportswriter).

When reading the current statistics for the season, that is, for the second read operation in Figure 8, the statistician also wants strong consistency. If an old statistic is returned, then the updated value written back will undercount the team's total runs. Since the statistician is the only person who writes statistics into the data store, she can use the read my writes guarantee to get the latest value (as discussed previously).

Stat watcher. Others who periodically check on the team's season statistics are usually content with eventual consistency. The statistical data is only updated once per day, and numbers

Table 4. Read guarantees for baseball participants.

Official scorekeeper	Read My Writes
Umpire	Strong Consistency
Radio reporter	Consistent Prefix & Monotonic Reads
Sportswriter	Bounded Staleness
Statistician	Strong Consistency, Read My Writes
Stat watcher	Eventual Consistency

that are slightly out of date are okay. For example, a fan inquiring about the total number of runs scored by his team this season, as shown in Figure 9, can perform an eventual consistency read to get a reasonable answer.

Conclusion

Clearly, storing baseball scores is not the killer application for cloud storage systems. And we should be cautious about drawing conclusions from one simple example. But perhaps some lessons can be learned.

Table 4 summarizes the consistency guarantees desired by the variety of baseball participants that were discussed in the previous section. Recall that the listed consistencies are not the only acceptable ones. In particular, each participant would be okay with strong consistency, but, by relaxing the consistency requested for his reads, he will likely observe better performance and availability. Additionally, the storage system may be able to better balance the read workload across servers since it has more flexibility in selecting servers to answer weak consistency read requests.

These participants can be thought of as different applications that are accessing shared data: the baseball score. In some cases, such as for the scorekeeper and sportswriter, the reader, based on application-specific knowledge, knows he can obtain strongly consistent data even when issuing a weakly consistent read using a read my writes or bounded staleness guarantee. In some cases, such as the radio reporter, multiple guarantees must be combined to meet the reader's needs. In other cases, such as the statistician, different guarantees are desired for reads to different data objects.

I draw four main conclusions from this exercise:

▸ **All of the six presented consistency guarantees are useful.** Observe that each guarantee appears at least once in Table 4. Systems that offer only eventual consistency would fail to meet the needs of all but one of these clients, and systems that offer only strong consistency may underperform in all but two cases.

▸ **Different clients may want different consistencies even when accessing the same data.** Often, systems bind a specific consistency to a particular dataset or class of data. For example, it is generally assumed that bank data must be strongly consistent while shopping cart data needs only eventually consistency. The baseball example shows that the desired consistency depends as much on who is reading the data as on the type of data.

▸ **Even simple databases may have diverse users with different consistency needs.** A baseball score is one of the simplest databases imaginable, consisting of only two numbers. Nevertheless, it effectively illustrates the value of different consistency options.

▸ **Clients should be able to choose their desired consistency.** The system cannot possibly predict or determine the consistency that is required by a given application or client. The preferred consistency often depends on how the data is being used. Moreover, knowledge of who writes data or when data was last written can sometimes allow clients to perform a relaxed consistency read, and obtain the associated benefits, while reading up-to-date data.

The main argument often expressed against providing eventual consistency is that it increases the burden on application developers. This may be true, but the extra burden need not be excessive. The first step is to define consistency guarantees developers can understand; observe that the six guarantees presented in Table 1 are each described in a few words. By having the storage system perform write operations in a strict order, application developers can avoid the complication of dealing with update conflicts from concurrent writes. This leaves developers with the job of choosing their desired read consistency. This choice requires a deep understanding of the semantics of their application, but need not alter the basic structure of the program. None of the code snippets that were provided in the previous section required any additional lines to deal specifically with stale data.

Cloud storage systems that offer only strong consistency make it easy for developers to write correct programs but may miss out on the benefits of relaxed consistency. The inherent trade-offs between consistency, performance, and availability are tangible and may become more pronounced with the proliferation of geo-replicated services. This suggests that cloud storage systems should at least consider offering a larger choice of read consistencies. Some cloud providers already offer two both strongly consistent and eventually consistent read operations, but this article shows their eventual consistency model may not be ideal for applications. Allowing cloud storage clients to read from diverse replicas with a choice of several consistency guarantees could benefit a broad class of applications as well as lead to better resource utilization and cost savings. C

References

1. Abadi, D. Consistency tradeoffs in modern distributed database system design. *IEEE Computer*, (Feb. 2012).
2. Amazon. Amazon DynamoDB; http://aws.amazon.com/dynamodb/.
3. Anderson, E., Li, X., Shah, M., Tucek, J. and Wylie, J. What consistency does your key-value store actually provide? In *Proceedings of the Usenix Workshop on Hot Topics in Systems Dependability*, (2010).
4. Bailis, P., Venkataraman, S., Franklin, M., Hellerstein, J. and Stoica, I. Probabilistically bounded staleness for practical partial quorums. In *Proceedings VLDB Endowment*, (Aug. 2012).
5. Brewer. E. CAP twelve years later: How the "rules" have changed. *IEEE Computer*, (Feb. 2012).
6. Calder, B. et. al. Windows azure storage: A highly available cloud storage service with strong consistency. In *Proceedings ACM Symposium on Operating Systems Principles*, (Oct. 2011).
7. Cooper, B., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.A., Puz, N., Weaver, D. and Yerneni, R. PNUTS: Yahoo!'s hosted data serving platform. In *Proceedings International Conference on Very Large Data Bases*, (Aug. 2008).
8. Google. Read Consistency & Deadlines: More Control of Your Datastore. Google App Engine Blob, Mar. 2010; http://googleappengine.blogspot.com/2010/03/read-consistency-deadlines-more-control.html.
9. Kraska, T., Hentschel, M., Alonso, G. and Kossmann, D. Consistency rationing in the cloud: Pay only when it matters. In *Proceedings International Conference on Very Large Data Bases*, (Aug. 2009).
10. Saito, Y. and Shapiro, M. Optimistic replication. *ACM Computing Surveys*, (Mar. 2005).
11. Terry, D., Demers, A., Petersen, K., Spreitzer, M., Theimer, M., and Welch, B. Session guarantees for weakly consistent replicated data. In *Proceedings IEEE International Conference on Parallel and Distributed Information Systems*, (1994).
12. Vogels, W. Eventually consistent. *Commun. ACM*, (Jan. 2009).
13. Wada, H., Fekete, A., Zhao, L., Lee, K. and Liu, A. Data consistency properties and the trade-offs in commercial cloud storages: The consumers' perspective. In *Proceedings CIDR*, (Jan. 2011).

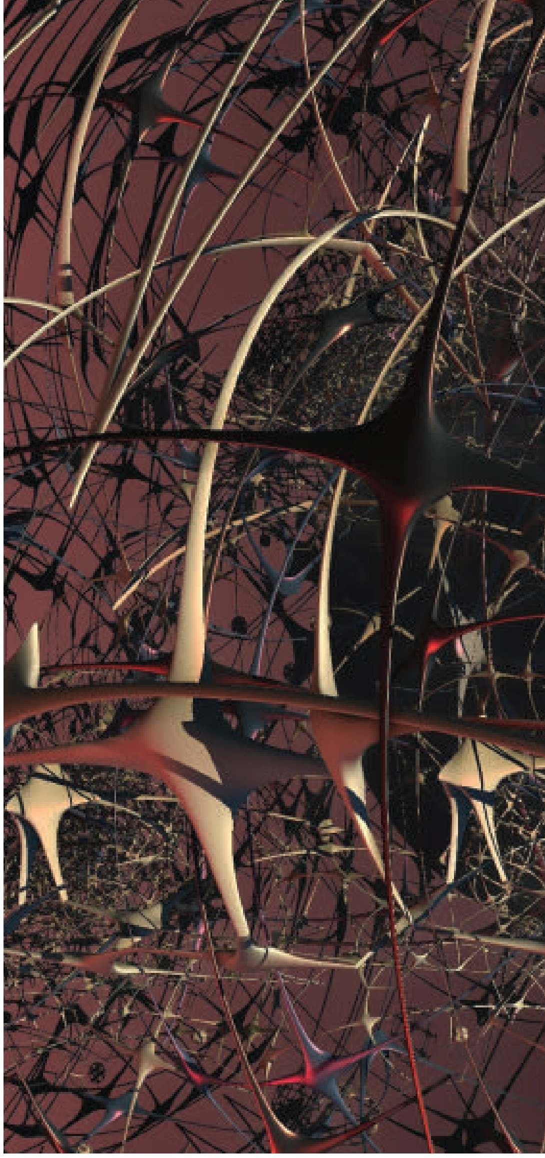
Doug Terry (terry@microsoft.com) is a Principal Researcher in the Microsoft Research Silicon Valley Lab, Mountain View. CA.

DOI:10.1145/2500501

'Where's' in a name?

BY GARETH TYSON, NISHANTH SASTRY, RUBEN CUEVAS, IVICA RIMAC, AND ANDREAS MAUTHE

A Survey of Mobility in Information-Centric Networks

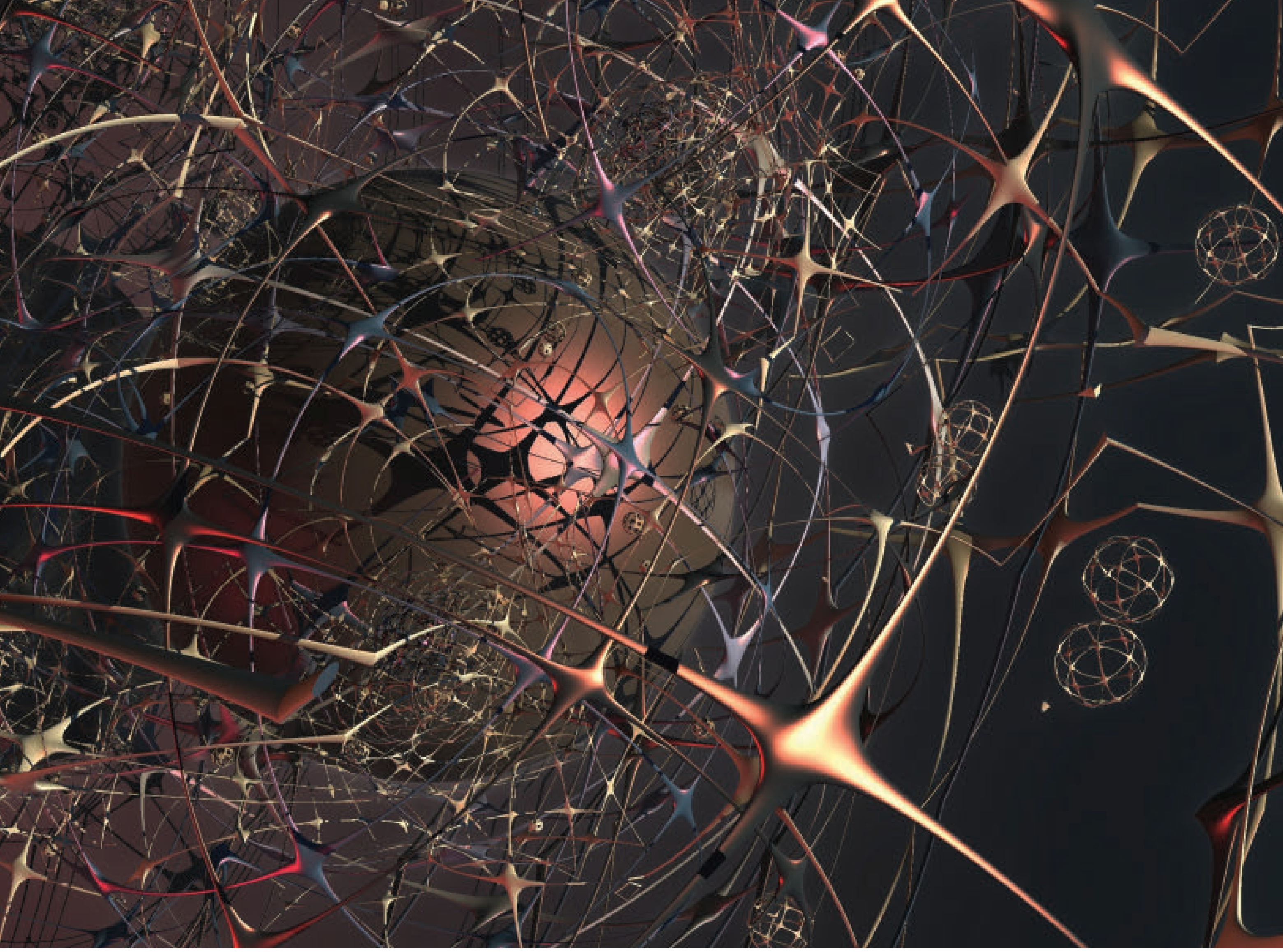


HOST MOBILITY HAS been a long-standing challenge in the current Internet architecture. Huge proportions of traffic are now attributed to mobile devices;[2] however, despite this prominence, mobility often remains a badly handled concept. Some have recently argued that the main reason for this lies in its choice of what to name.[14] The Internet Protocol (IP) names hosts based on their topological network location. Through this, it intrinsically binds the *what* (the name) to the *where* (the address). Consequently, a mobile host moving its physical location is often required to change its name creating numerous problems.

Observations such as this have led to a flurry of research looking at how the future Internet could be redesigned. A prominent example is that of information-centric networks (ICNs).[14,26,30] ICNs propose a key paradigm shift, which involves replacing the Internet's existing host-based naming scheme with an information-based one instead. This article there-

» key insights

- **Two of the most promising characteristics of the future Internet will be an increased focus on content delivery and ubiquitous host mobility.**
- **Information-centric networks have been proposed as a powerful architecture for better supporting host mobility in this context. By removing the concept of location, it is intended that networks are liberated from maintaining complex location-based information about nodes during mobility.**
- **Despite its attractiveness, there are still many open questions that remain to be answered in the field. These include questions relating to handling publisher mobility, routing information back to mobile requesters, and dealing with real-time streams.**



fore chooses to follow Shakespeare's advice and ask "What's in a name?" rather than IP's approach of "Where's in a name?"

Through this principle, an ICN becomes an infrastructure that revolves around the provision of uniquely identified content[a] to consumers, rather than the routing of data between device pairs. By removing the use of host-centric naming, it is therefore hoped that it will be possible to seamlessly change a host's physical and topological location without needing to perform the types of complex network management that host-centric networks require (for example, creating forwarding between home and foreign addresses).[22]

In this article, we aim to explore and review these concepts and ideas. We first explore what an ICN is, before investigating some of the key benefits of designing a network around the concept of information. From this, we then present some prominent ICN proposals before using these to identify important remaining challenges.

Defining an ICN

In essence, an ICN is a network that has the primary purpose of distributing information. As such, it exposes a content request style abstraction unlike the existing Socket API. This is because a host-centric network (for example, the Internet) is designed to route packets from a source to a destination, while an information-centric network is designed to deliver information from a provider to a consumer.

Its roots lie in previous attempts to build infrastructures centered on the dissemination of information. Most noteworthy are publish/subscribe mechanisms,[7] as well as peer-to-peer content delivery systems.[19] Both use overlay architectures to allow publishers to make information (for example, data, files, and so on) available to consumers. Importantly, however, these various systems are disparate with applications typically utilizing specific infrastructures and protocols for their own needs.[29] In contrast, an ICN attempts to underpin these applications through the ubiquitous support of information dissemination as an explicit network layer concept, rather than something simply built over it. Consequently, ICNs introduce new network components that unify such things as information naming, routing, security and management within a single architecture. Through this, a number of key differences between traditional host-centric networking can be identified:

- *Naming:* Host-centric networks utilize names that identify a host by its topological position. In contrast ICNs name unique items of content,[10] which could exist in many places throughout the network.
- *Routing:* Host-centric networks route between hosts using pairs of topological identifiers (for example, IP

a We will use the terms information and content interchangeably.

ILLUSTRATION BY MARALOVA

addresses). In contrast, ICNs route (or bind) between points of consumption and optimal content sources.

▸ *Security:* Host-centric networks attempt to secure communication channels between hosts. In contrast, ICNs attempt to secure the integrity of individual content objects, regardless of their delivery mechanism.

▸ *API:* Host-centric networks expose APIs that allow data to be sent to a given location. In contrast, ICNs expose APIs that allow content to be published and consumed.

In the remainder of this article, we explore the benefits of the differences mentioned here, specifically from the viewpoint of improving node mobility.

The concept of an ICN does not force applications to take on host-centric information. Instead, it detaches the application from such concerns.

What Are the Benefits of ICN for Mobility?

The proposed improvement in mobility support is achieved by refocusing network routing on content objects, rather than hosts. Consequently, in an ICN, changes in a node's physical location do not necessarily need changes in its related network information (for example, routing state). This high-level concept therefore opens up many potential benefits. Here, we look at some of the possible advantages that could be gained if the theoretical principles of ICN were realized.

Host multihoming. A long-standing challenge in host-centric networks is allowing mobile hosts to exploit multiple network interfaces (for example, Bluetooth, UMTS, Wi-Fi, among others). This is because typically most protocols rely on establishing individual connections using each host's address. However, because an address is bound to a specific network interface, it is difficult to easily switch between them. For example, a HTTP GET request is always received over a single TCP connection from a single source address. Consequently, during mobile hand-offs, it is difficult to exploit multiple potential network interfaces that might be available when using HTTP.

In contrast, the concept of an ICN detaches itself from host-to-host connections. Instead, communications within an ICN are typically based around a request/reply model. As such, requests can easily be multiplexed over multiple interfaces. This means that applications running on a multihomed ICN node could seamlessly exploit these different interfaces without needing to understand which interface has actually been used.

Network address consistency. Currently, many mobility mechanisms attempt to maintain consistency in the node's network address. This is vital for many applications that may utilize a node's IP address for long-term usage. A typical example is BitTorrent, which will see a node's IP address being registered with a tracker for future discovery. Mobile IP,[22] for instance, introduces the concept of a Home Agent to allow hosts to change their physical address, while still maintaining a constant public address. This, however, creates undesirable overheads due to the need to tunnel data through this Home Agent. Unfortunately, the alternative requires greater intelligence in applications to make them aware of mobility, thereby allowing them to update their location information.

In contrast, the concept of an ICN does not force applications to take on host-centric information. Instead, it detaches the application from such concerns. This allows the application to abstractly publish or consume content, without the need to store (or even know) its own network-layer address. In essence, it promotes content, which is already an explicit application-layer element, to an explicit network-layer entity as well, thereby requiring the application to only maintain knowledge that does not deviate from its own traditional knowledge base.

Removal of connection-oriented sessions. A key problem with mobility in host-centric networks is their frequent dependency on connection-oriented protocols (for example, TCP). Thus, mobility can often require the reestablishment of these connection-oriented sessions so that both parties are aware of the up-to-date network addresses, as well as any pertinent parameters. Generally, TCP sessions are used in host-centric networks to establish reliability parameters (for example, sequence numbers) and configure flow/congestion control (for example, window size). This is necessary because the network stack does not have an explicit understanding of the data it is sending/receiving, therefore requiring bilateral cooperation to ensure a receiver obtains the right data at an appropriate speed.

In contrast, in an ICN, communications are made explicit within the network stack: when a node sends a request for an object, it can understand if that request has been satisfied. As such, the communications model becomes receiver-driven, without the need for cooperation from the sender to achieve in-order reliability. Through this, it also becomes possible to perform flow/congestion control by simply altering the frequency of requests. Therefore, sessions established between specific parties become less necessary.

Scoping of content and location. Currently, consumers are generally identified by their location (IP address). Often, however, this is incorrectly used for scoping purposes. That is, incorrect information is interpreted from the address. For instance, the BBC iPlayer service can only be accessed from U.K. IP addresses; consequently, this makes mobility difficult for legitimate U.K. residents who may temporarily utilize connectivity abroad. A similar problem emerges in CDNs when attempting to utilize IP addresses for selecting optimal content replicas. This is because (at request time) the CDN will utilize a node's location to resolve an optimal source, even though the node may later change its location.

In contrast, an ICN makes an explicit separation between the *what* (the user or content) and the *where* (their location). Thus, a node's location can seamlessly change while still maintaining a consistent name (and profile) for the user. Through this, it would not be necessary to (incorrectly) interpret things from changing location-based addresses; instead, such information could be encapsulated within separate node descriptions that the network could then exploit (for example, for access control).

Resilience through replication. Information exchange in a host-centric network is usually based on some concept of location (for example, a URL). As such, if the host identified in the URL fails or, alternatively, if any of the intermediate routers fail, the content will become unavailable (this is particularly prevalent in MANETs[6] and DTNs[27]).

In contrast, an ICN does not bind content to specific locations through the use of host identifiers; instead, content is the key addressable entity. This allows content to be stored anywhere, potentially allowing local copies to be retrieved. On the one hand, this can improve performance.[28] However, beyond this, the effects of network failures can also be mitigated.[31] This is because ICN caching can increase the number of potential end points for each request, thereby adding redundancy.

Information-Centric Proposals

Here, we briefly review some prominent ICNs, alongside their approaches to handling mobility.

NDN[14] is a prominent design (also known as CCN and CCNx). Figure 1 provides an overview of its operation. Content naming is based on a flexible hierarchical structure, allowing a variety of namespaces. In NDN, a content request is issued by sending an Interest packet, which is routed through the network to an instance of the content.

Figure 1. Overview of NDN.

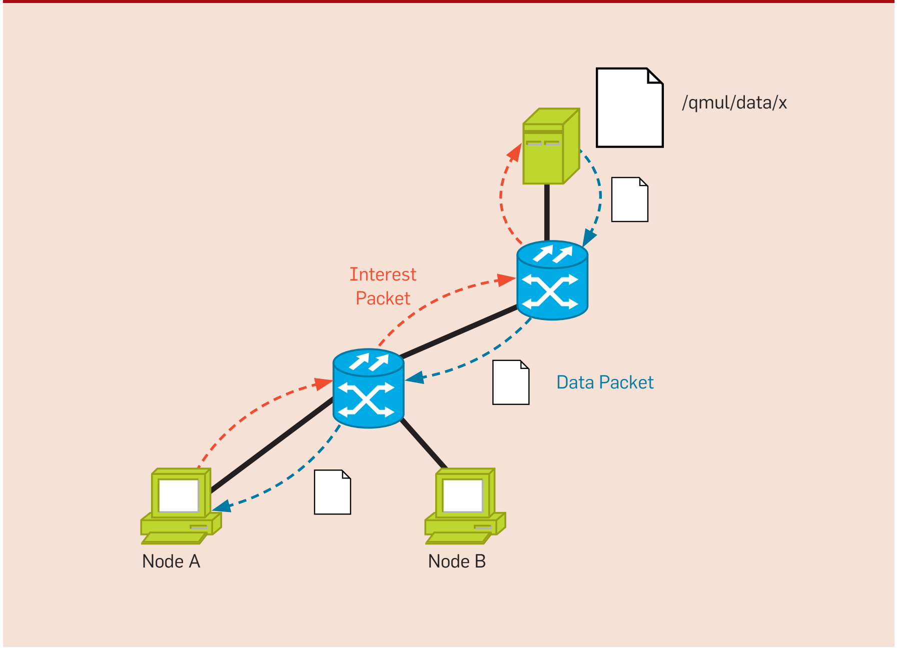
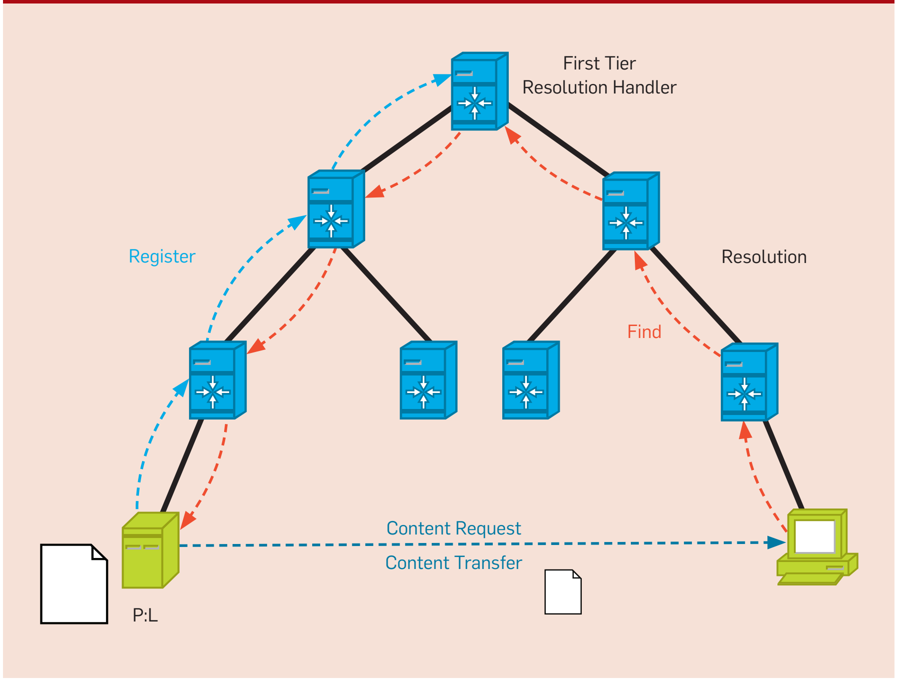


Figure 2. Overview of DONA.



Routing is performed using similar mechanisms to current IP infrastructure, utilizing longest prefix matching. Therefore, to maintain scalability, the naming hierarchy is exploited to aggregate address space together in routing tables. Thus, in NDN each request is only resolved to a specific location during the final stages of the routing process (that is, at the last hop). Following this, if available, the source responds with a data packet, which follows the reverse path back to the requester using "breadcrumbs" left in a Pending Interest Table on each router (the data packets are also cached on each router). Requests are therefore performed on packet-sized objects.

Consumer mobility in NDN is intrinsic due to its consumer-driven nature. When a consumer relocates, it can reissue any previously sent Interest packets that have not been satisfied yet. This can occur seamlessly because there is no need to perform any new registrations (although resending Interests obviously has overheads). Through this, it has been shown that NDN can still handle up to 97% of requests even during high mobility.[32] Provider mobility, however, is more challenging as, practically speaking, there is no separation between content identifier and routing locator. As such, to ensure route aggregation, the content item's naming hierarchy must also reflect the underlying topology that it is routed over. Moving individual content items to different locations could therefore undermine this aggregation and create a state explosion in the core of the network. Consequently, it is better for domains of content objects to move as one, rather than having individual items of content move independently.

DONA[18] introduces ICN in the form of a replacement (or supplement) to DNS. Content names are of the form P:L, where P is the cryptographic hash of the publisher's public key and L is a label that identifies the content. DONA requires each domain to deploy servers called Resolution Handlers (RH) that index content stored by authorized storage points. RHs are then structured into a tree topology that represents the BGP topology of the network, as shown in Figure 2. Lookups are performed by querying a consumer's local RH; if no reference is found, the query is forwarded up the tree until a source is discovered. The request is then forwarded to the source and an out-of-band delivery is established by the source. Importantly, however, due to the overhead of routing each request, DONA is unlikely

Figure 3. Overview of NetInf.

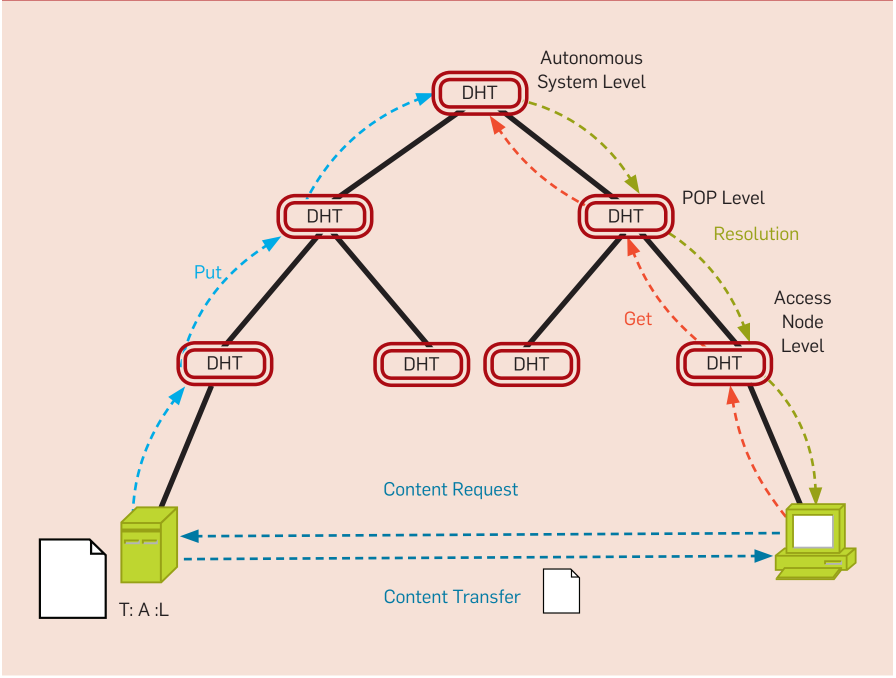
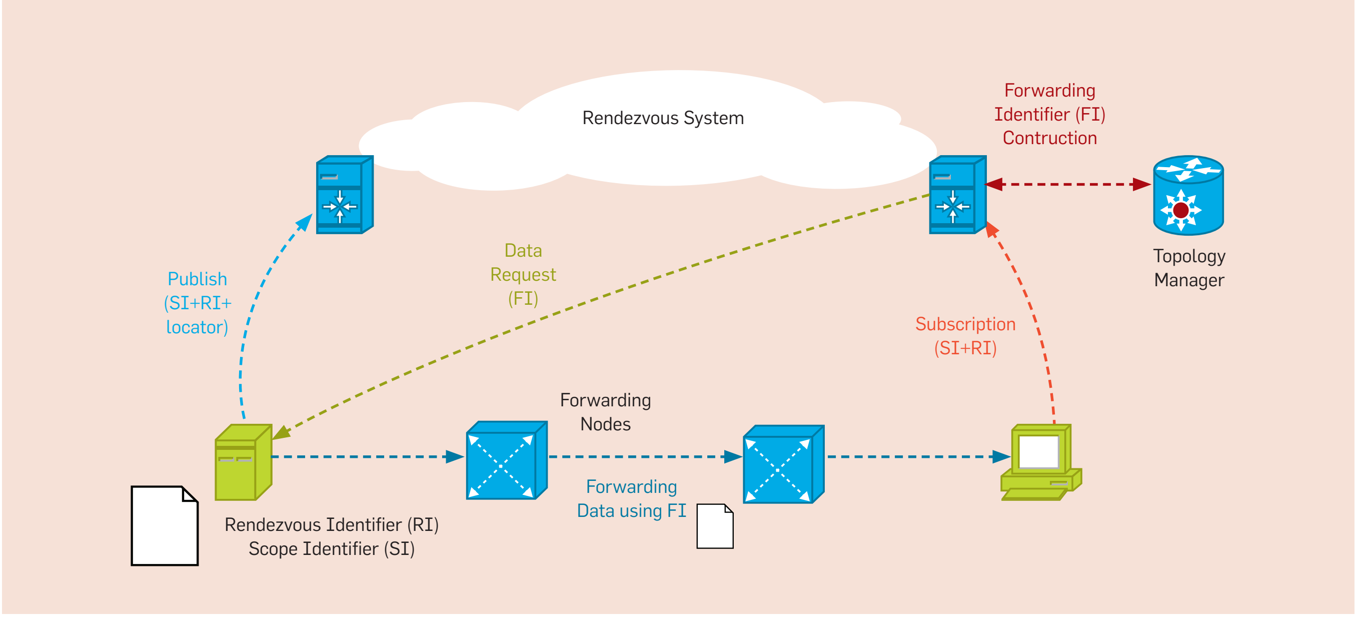


Figure 4. Overview of PURSUIT.



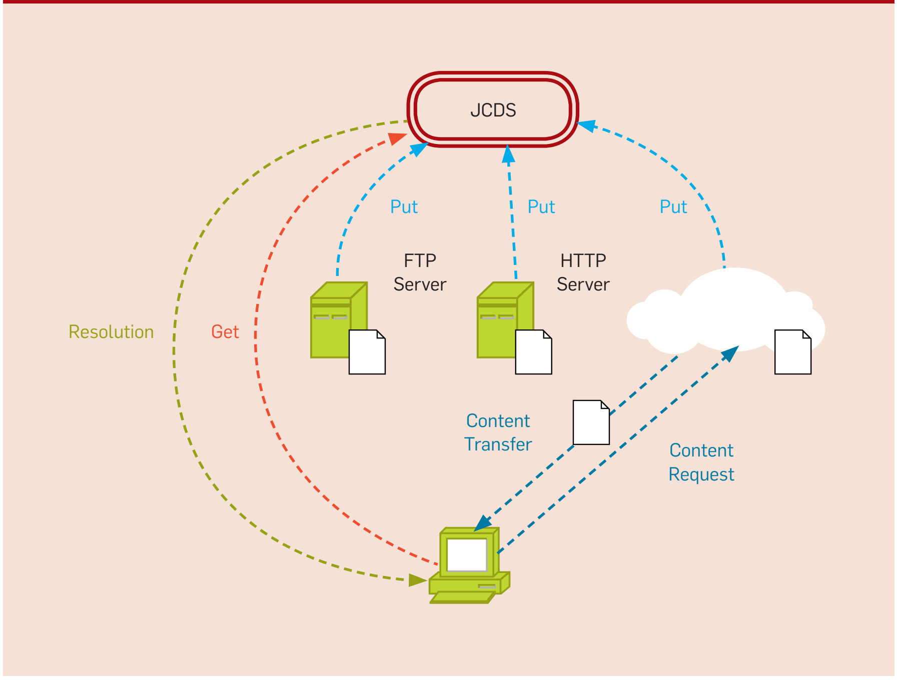
to use packet-sized objects like NDN does (to minimize load).

DONA handles consumer mobility by changing a host's RH to that of the new network. If necessary, any existing requests can then simply be reissued to the new RH to locate the new optimal source. Unlike some other designs, however, DONA relies on out-of-band deliveries of content; although not stipulated, this would likely take place over TCP/IP. Consequently, unlike NDN, this would require session reestablishment after consumer relocation (either to reestablish the same connection or one with a newly selected source), thereby complicating the process. Allowing nodes to republish their content with the new network's RH also supports provider mobility. Clearly, however, any active transfers in this situation would then need to be re-established by the consumer or, alternatively, continued using a mechanism like Mobile IP.

NetInf primarily relies on a Name Resolution (NR) service. Using the NR service, providers publish Named Data Objects (NDOs) alongside their locators (termed routing hints) for later discovery by consumers or intermediate routers forwarding requests.[5] The resolution process therefore simply maps each NDO's self-certifying identifier[4] to a set of locators. A Multi-level DHT (MDHT),[3] allowing global content lookups while also supporting local resolution, underpins the NR service. This process is shown in Figure 3 with recursive querying of the MDHT. Requesters therefore perform lookups, which are responded to with either a list of potential sources or a selected optimal source. Content can then be delivered from a source using any supported delivery protocol, including those that allow in-router caching on the intermediate path (for example, Ott et al.[21]).

Consumer mobility in NetInf is achieved through its indirection between identifiers and locators. The exact details of this vary based on the chosen locator selector mode.[3] In the requester-controlled mode, a requester is provided with a list of potential sources, thereby allowing a node to select a new optimal source following relocation. In contrast, the MDHT-controlled mode results in a consumer

Figure 5. Overview of Juno.



only receiving a single source on each request, mandating a relocated node to contact the NR service again. Regardless of this, both modes should enable mobility, assuming fast lookups. Provider mobility is more challenging, as it requires the NR service to be updated and all consumers to rebind to the new location; however, it is claimed that updates can be scalably handled. It is important to note, however, that this would only maintain content availability for new requests—existing requests would either need to be re-sent or continued though a mechanism similar to Mobile IP (as with DONA).

PURSUIT[26] proposes the use of a publish/subscribe abstraction, as opposed to the synchronous get used by most other approaches. Within PURSUIT, significant focus is given to the decomposition of network functionality into three key components: Rendezvous, Topology Management, and Forwarding. Each of these could be potentially implemented in different ways, however, here we focus on their current realization for global networking. When providers wish to publish content, they register it with the Rendezvous System using both a Scope and Rendezvous Identifier (SI and RI). These are identifiers that are interconnected by a tree structure, in which SIs are inner nodes that aggregate RIs together (as leaf nodes). The Rendezvous System is therefore a lookup service (for example, a DHT) that can map an identifier to a data source, as shown in Figure 4. Once a source is discovered, the Topology Manager is used to construct a path to the source, which then results in a Forwarding Identifier (FI) being generated. Within the prototype, the FI is a bloom filter, which encodes the hops that any data must take through the network to reach the subscriber from the provider, that is, source routing (LIPSIN[15]).

Consumer mobility in PURSUIT is relatively straightforward to achieve. When a consumer relocates, it resubscribes to the content being accessed. This results in a new FI being computed for the host's new location. Clearly, the efficiency of consumer mobility is therefore dependent on the speed at which new FIs can be generated and mapped to RI/SIs. To alleviate this, the architecture proposes the use of explicit caches that providers can continue to stream to while consumers are switching between access points.[11] It is claimed that PURSUIT can lead to 50% less packet loss during mobility compared with Mobile IPv6.[8] Provider mobility would have a higher overhead as it would require updating information in the Rendezvous System. More importantly, it would also invalidate the existing FIs a provider was using. Consequently, new routes would need to be computed for all subscribers. The speed of this process would therefore largely define the hand-off delay; it is important to note, however, that relocation within the same domain would allow the majority of the existing path to be reused, thereby increasing speed.

Juno[30] proposes the placement of information-centric functionality in the middleware layer, shown in Figure 5. Content is based on self-certifying identifiers that are indexed on a DHT called the Juno Content Discovery Service (JCDS). Content identifiers are therefore resolved rather than routed to. Unlike other designs, however, Juno focuses on achieving backward compatibility by performing software reconfiguration to interoperate with any sources that might offer the content, regardless of their delivery protocols. To achieve this, Juno attempts to discover as many content sources as possible by also probing third party indexing services such as eMule. Once a set of sources have been discovered, Juno's Delivery Framework retrieves the content by utilizing dynamically attachable protocol plug-ins that each has the capability to interact with a given source/protocol. For instance, if a HTTP source were located, a HTTP plug-in would be dynamically attached to retrieve the content. Importantly, Juno attempts to intelligently reconfigure between the use of different sources based on the higher-level needs of the application (for example, performance, resilience, monetary cost, and so on.)

Consumer mobility is easily achieved in Juno by simply reselecting sources after host relocation. This can be done locally as Juno keeps a full list of sources from the resolution process. The hand-off delay, however, will be defined by the bootstrap time of the delivery protocols being used; for example, connecting to a BitTorrent swarm will take longer than establishing a HTTP connection. Provider mobility is similarly possible by simply updating the JCDS; like NetInf, the performance of this depends on the DHT.

Research Challenges

Clearly, an increasingly large research effort is being invested in ICN. However, a number of key challenges remain, particularly in the mobility domain. The most prominent of these challenges include the following:

Provider mobility. Broadly speaking, consumer mobility is a well-handled phenomenon due to the consumer-driven nature of most ICN designs. However, a larger challenge is maintaining routing consistency during provider mobility. This is because whenever a provider relocates, it is clearly necessary to update (potentially global) locator information. This is heavily exacerbated by the obvious increase in the number of content objects when compared to hosts (an ICN must be able to deal with at least 10^{12} objects[9]). The effects of this can be mitigated via caching and replication but less frequently requested content is still likely to suffer if high-speed provider hand-offs cannot be achieved.

The precise focus of this challenge varies with the different naming and discovery techniques employed. NDN, for instance, uses hierarchical naming and route aggregation to improve scalability. However, because names are also used for routing locators, they must efficiently map to topological locations. This creates significant challenges when relocating providers to different topological positions because it clearly undermines the hierarchy of the address space; in fact, using any content that is cached off-path introduces a similar challenge. Unfortunately, line-speed switching relies heavily on this aggregation, meaning that provider mobility will introduce significant scalability challenges. Regardless of this, clearly, routing information will need to be disseminated during provider mobility leading to convergence delays.

Challenges also arise in resolution approaches such as NetInf or Juno. This is because any provider mobility must be reported to the resolution service; clearly, high levels of mobility could result in phenomenal loads. For instance, in D'Ambrosio et al.,[3] the authors discuss the handling of 1% of churn in registrations, however, mobility could greatly increase this percentage. Similarly, when this occurs, potentially all related consumers will need to be notified of changes to allow them to rebind to the new source via the resolution service (opposed to NDN, which does not require this). This is particularly problematic if subsets of larger objects cannot be independently requested, leading to the need to request the whole object again. As previously mentioned, to address this, approaches similar to Mobile IP have begun to emerge, allowing messages to be forwarded to a mobile provider's current location[13] (thereby not requiring resolution/routing updates in the core). Clearly, these solutions reintroduce some of the problems (for example, tunneling overheads) that ICNs wished to move away from. A key point to observe, however, is that during provider mobility, it becomes possible for consumers to rebind to alternate sources (for example, a cache), thereby mitigating hand-off delays. Despite this, a prominent remaining challenge is to design mechanisms that can elegantly allow provider mobility without the need for any complicated or high overhead procedures.

Response routing. Due to the (attempted) removal of location from the concept of ICN, many approaches utilize predefined pairwise hop-by-hop knowledge (for example, breadcrumbs) to ensure data can find its way back to consumers without needing host-centric routing. Unfortunately, however, this can be challenging in a mobile network because paths could change frequently. In NDN, for instance, data packets always follow the reverse paths of their equivalent Interest packets; thus, if a host changes its location, the response path will change, leaving a window of potentially many data packets being routed to an out-of-date location (in TCP, for instance, window scaling allows windows of approximately a gigabyte[33]). Interestingly, some protocols designed for handling this network dynamism[25] still rely on reverse path routing. Alternatively, other solutions avoid multi-hop routing and simply rely on one-to-one opportunistic connections,[12] thereby losing access to content that is multiple hops away. A related situation also arises in PURSUIT through its use of per-hop source routing. If, for instance, the computed route changes due to mobility, this per-hop knowledge will become outdated. Encoding redundant virtual links[15] can mitigate this but even these could become invalid due to mobility. Clearly, if ICNs are to be deployed in mobile environments, handling these physical path changes is an extremely important research issue to address. This will therefore likely involve creating a compromise between both host-centric and information-centric routing.

Discovering local cached content. One of the key benefits of an ICN is the ability to deploy ubiquitous caching. This, however, can create significant challenges in mobile environments, particularly MANETs and DTNs, due to the potential cost of managing cached replicas.

The specifics of this challenge will vary with the particular approach employed. Approaches such as NDN would likely suffer heavily due to the increased overhead of maintaining routing information for cached content sources. In fact, to mitigate this, some recent mobile routing algorithms (for example, CHANET[1]) do not require mobile nodes to advertise their cached content, instead only allowing opportunistic on-path caching. In contrast, approaches such as DONA and Juno, which utilize resolution services, would also suffer due to the need for resolution updates for every cached instance (this is analogous to extremely high levels of provider mobility). Further, when considering ad hoc environments, things could be even worse if mobility meant these resolution services somehow became inaccessible. Interestingly, promising information-centric MANET routing protocols have begun to emerge, addressing some problems (for example, LFBL[20] and Slinky[17]). However, these are still yet to be extensively tested in terms of performance and scalability, among others.

A prominent research challenge that therefore remains is to build and evaluate naming, resolution and routing schemes that can handle this type of unpredictable ad hoc relocation of content. A particularly important challenge is achieving this for unpopular content that does not benefit from caching (that is, accessed only once). For instance, when dealing with smaller MANETs (for example, <300), Varvello et al.[31] found the performance benefits of using more sophisticated structured routing protocols (for example, GHT[23]) were dwarfed by their overheads due to the presence of unpopular content.

One of the key benefits of an ICN is the ability to deploy ubiquitous caching. This, however, can create significant challenges in mobile environments due to the potential cost of managing cached replicas.

Real-time hand-off delays. Mobility during time insensitive network interactions is a relatively easy issue to handle in many ways. This is because there is no real constraint on the hand-off delay. In contrast, mobility during real-time communications (for example, video conferencing) is far more difficult because hand-offs must be in the order of milliseconds.

Typically, the main benefit of using an ICN for mobility is that cached or replicated copies of the content could potentially mitigate any hand-off delays. However, many real-time communications have little potential for caching (for example, a voice call). Further, as some real-time communications are multidirectional, all parties behave as both consumers and providers, thereby increasing the network load of mobility. Practically speaking, systems that use content resolution rather than routing would likely perform better because a resolution service would only require a small number of centralized updates. However, the exact performance has yet to be understood. A prominent remaining challenge is therefore ensuring real-time multimedia can be fully supported in an ICN. On the one hand, this refers to handling mobility, but it also extends to many other issues including QoS and QoE.

Privacy and security in open mobile systems has been a long-term challenge, with many possible attacks. Unsurprisingly, a key research challenge is therefore handling these types of concerns in an ICN. In principle, ICNs primarily focus on securing the content itself through its unique name; that is, guaranteeing a content item is what it claims to be. This approach, however, can obviously introduce privacy challenges because it requires nodes to expose their interests to the network. Thus, if third parties can map identifiers to content items (which often will be possible), a user's privacy could be heavily undermined.

Beyond this, alternate security issues relating to such things as routing are yet to be adequately explored. In theory, any node can publish the ability to serve an item of content, thereby empowering malicious nodes to manipulate routing. This can be an issue in traditional fixed infrastructure; however, it is particularly prevalent in networks such as MANETs, which have extremely open routing policies (for example, black hole routing).

Practical challenges. A further challenge that perhaps exceeds all others is the question of practical deployment. Clearly, the discussed benefits can only be gained if a node connects to (and moves between) domains that support ICN. Unfortunately, however, in practice, it is likely that any near-future ICN deployments will be incremental overlay structures, making it impossible to quantify the effectiveness of mobility support without concrete details of their configuration.

The existence of islands of ICNs connected via tunnels, for instance, could severely damage mobility support if nodes were required to 'dial-in' via VPN-like services after every handoff. Even low-delay access services within the domain of the node might create unacceptable delays if complex authentication were required. Interestingly, this problem becomes even more challenging when considering the diversity of ICN proposals, likely leading to the need for complicated inter-networking technologies.

Despite this, clearly, the long-term goal of ICN would be to move toward a native deployment (for example, dual stack). This would, of course, be an incremental process in which the benefits increase with each new domain's uptake (as with IPv6 deployment[16]). However, an intelligent deployment strategy could help mitigate any potential problems. Specifically, using the lessons learned from previous overlay technologies, many problems could be avoided by integrating underlay knowledge (for example, locality awareness[24]). This still does, however, leave open practical questions such as how nodes might discover and connect to ICN-enabled domains, including various autoconfiguration challenges.

Conclusion

This article has explored the world of ICN, looking at how a shift from host-centric to information-centric design principles could support greater mobility in the future Internet. Clearly, this is a hugely important topic, as we observe more traffic being generated by mobile hosts.[2] However, despite the clear potential of ICN, a number of challenges remain. Of particular importance is the ability to handle the scalability challenges of increasing numbers of content items (in the form of router entries) and providers (in the form of router caches). Whereas this is a significant problem in fixed infrastructure, it is even more challenging in mobile environments.

It is important, however, to note that these are not necessarily weaknesses in ICN. Instead, they are exciting topics deserving future attention. Such promising research has already begun to develop, however, it is evident the diversity of mobile content access means that any one-size-fits-all approach will fall short. Consequently, we believe the key future research challenge is building flexible general-purpose architectures that can handle all the situations discussed within this article.

References

1. Amadeo, M. and Molinaro, A. CHANET: A content-centric architecture for IEEE 802.11 MANETs. In *Proceedings of the Intl. Conference on the Network of the Future* (2011).
2. Cisco visual networking index: Forecast and methodology, 2011–2016.
3. D'Ambrosio, M., Dannewitz, C., Karl, H. and Vercellone, V. MDHT: A hierarchical name resolution service for information-centric networks. In *Proceedings of the ACM SIGCOMM Workshop on ICN* (2011).
4. Dannewitz, C., Golic, J., Ohlman, B. and Ahlgren, B. Secure naming for a network of information. In *Proceedings of INFOCOM IEEE Conference on Computer Communications Workshops*. IEEE, 2010.
5. Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B. and Karl, H. Network of information (netinf) an information-centric networking architecture. *Computer Communications* (2013).
6. Djenouri, D., Khelladi, L. and Badache, A. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys & Tutorials 7*, 4 (2004).
7. Eugster, P., Felber, P., Guerraoui, R. and Kermarrec, A. The many faces of publish/subscribe. *ACM Computing Surveys 35*, 2 (2003), 114–131.
8. Fotiou, N., Nikander, P., Trossen, D. and Polyzos, G.C. Developing Information Networking Further: From PSIRP to PURSUIT. In *Proceedings of the Intl. Conference on Broadband Communications, Networks, and Systems* (2010).
9. Ghodsi, A., Koponen, T., Raghavan, B., Shenker, S., Singla, A. and Wilcox, J. Information-centric networking: Seeing the forest for the trees. In *Proceedings of the ACM Workshop on Hot Topics in Networks* (2011).
10. Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, J. and Shenker, S. Naming in content-oriented architectures. In *Proceedings of the ACM SIGCOMM Workshop on ICN* (2011).
11. Giannaki, V., Vasilakos, X., Stais, C., Polyzos, G.C. and Xylomenos, G. Supporting mobility in a publish subscribe internetwork architecture. In *Proceedings of the IEEE Symposium on Computers and Communications* (2011).
12. Helgason, O.R., Yavuz, E.A., Kouyoumdjieva, S.T., Pajevic, L. and Karlsson, G. A mobile peer-to-peer system for opportunistic content-centric networking. In *Proceedings of the ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds*, 2010.
13. Hermans, F., Ngai, E. and Gunningberg, P. Mobile sources in an information-centric network with hierarchical names: An indirection approach. In *Proceedings of the 7th Swedish National Computer Networking Workshop* (2011).
14. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., and Braynard, R.L. Networking named content. In *Proceedings of the 5th ACM CoNEXT* (2009).
15. Jokela, P., Zahemszky, A., Esteve Rothenberg, C., Arianfar, S. and Nikander, P. LIPSIN: Line speed publish/subscribe inter-networking. *SIGCOMM Comput. Commun. Rev. 39*, 4 (2009), 195–206.
16. Karpilovsky, E., Gerber, A., Pei, D., Rexford, J. and Shaikh, A. Quantifying the extent of IPv6 deployment. In *Proceedings of the Passive and Active Network Measurement Conference* (2009).
17. Kawadia, V., Riga, N., Opper, J. and Sampath, D. Slinky: An adaptive protocol for content access in disruption-tolerant ad hoc networks. In *Proceedings of the MobiHoc Workshop on Tactical Mobile Ad Hoc Networking* (2011).
18. Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K.H., Shenker, S. and Stoica, I. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev. 37*, 4 (2007), 181–192.
19. Lua, E., Crowcroft, J., Pias, M., Sharma, R. and Lim, S. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials 7*, 2 (2005), 72–93.
20. Meisel, M., Pappas, V. and Zhang, L. Ad hoc networking via named data. In *Proceedings of MobiArch* (2010).
21. Ott, J., Budigere, K., Sarolahti, P. and Perkins, C. Poor man's content centric networking (with TCP). Tech. Rep. Aalto-ST 5/2011. Aalto University, 2011.
22. Perkins, C.E. and Myles, A. Mobile IP. In *Proceedings of the Intl. Telecommunications Symposium* (1994), 415–419.
23. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. and Shenker, S. GHT: A geographic hash table for data-centric storage. In *Proceedings of the ACM Workshop on Wireless Sensor Networks and Applications* (2002).
24. Rumin, R., Laoutaris, N., Yang, X., Siganos, G. and Rodriguez, P. Deep diving into bittorrent locality. In *Proceedings of IEEE INFOCOM* (2011).
25. Soon-Young Oh, M.G. and Lau, D. Content centric networking in tactical and emergency MANETs. In *Proceedings of the IFIP Wireless Days Conference* (2010).
26. Trossen, D. and Parisis, G. Designing and realizing an information-centric Internet. *IEEE Communications Magazine 50* (July 2012).
27. Tyson, G., Bigham, J. and Bodanese, E. Towards an information-centric delay-tolerant network. In *Proceedings of the IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking* (2013).
28. Tyson, G., Kaune, S., Miles, S., El-Khatib, Y., Mauthe, A. and Taweel, A. A trace-driven analysis of caching in content-centric networks. In *Proceedings of the Intl. Conference on Computer Communication Networks* (2012).
29. Tyson, G., Mauthe, A., Kaune, S., Grace, P. and Plagemann, T. Juno: An adptive delivery-centric middleware. In *Proceedings of the 4th Intl. Workshop on Future Media Networking* (2012).
30. Tyson, G., Mauthe, A., Kaune, S., Grace, P., Taweel, A. and Plagemann, T. Juno: A middleware platform for supporting delivery-centric applications. *ACM Transactions on Internet Technology* (2012).
31. Varvello, M., Rimac, I., Lee, U., Greenwald, L. and Hilt, V. On the design of content-centric MANETs. In *Proceedings of the Intl. Conference on Wireless On-Demand Network Systems and Services* (2011).
32. Wang, J., Wakikawa, R. and Zhang, L. DMND: Collecting data from mobiles using named data. In *Proceedings of the IEEE Vehicular Networking Conference* (2010).
33. Wolfgang, J. and Tafvelin, S. Analysis of internet backbone traffic and header anomalies observed. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement* (2007).

Gareth Tyson (gareth.tyson@eecs.qmul.ac.uk) is a senior research associate at Queen Mary, University of London, U.K.

Nishanth Sastry (sastry@kcl.ac.uk) is a lecturer at King's College London, U.K.

Ruben Cuevas (rcuevas@it.uc3m.es) is an assistant professor at Universidad Carlos III de Madrid, Spain.

Ivica Rimac (ivica.rimac@alcatel-lucent.com) is a principal investigator at Alcatel-Lucent Bell Labs, Stuttgart, Germany.

Andreas Mauthe (andreas@comp.lanc.as.uk) is a senior lecturer at Lancaster University, U.K.

DOI:10.1145/2534706.2534725

Technical Perspective The Cleanest Garbage Collection

By Eliot Moss

GARBAGE COLLECTION, THE quirky name used for automatic storage management, might well be called memory recycling if it were invented today. Indeed, it has a venerable history, nearly as long as that of computing itself, extending back to early LISP implementations, with papers appearing from 1960 onward. GC, as it is affectionately known, also developed a reputation for being slow and requiring a lot more memory than explicitly managed memory. If that was not enough, most GC algorithms would wait for the designated memory to fill, then stop the program, collect, and restart the program, introducing pauses into the primary computation.

Proponents of GC have persuasive software engineering arguments on their side: automatic storage reclamation simplifies programming and reduces errors, thus increasing programmer productivity. Ultimately, increasing computer speed and memory capacity made GC a reasonable choice for a much wider range of systems. Java brought it into the mainstream, where it has been quite successful. We have even reached the point where C++, originally posted with signs reading "No GC here!", now offers optional support for it.

But the holy grail for automatic storage management has been to achieve GC for *real-time* systems. Real-time GC is challenging for several reasons. One is that real-time systems cannot tolerate large pauses. This requires the collector either to be incremental at a fine grain or to run concurrently with the program (called the *mutator* in GC parlance because it nefariously changes pointers around while the GC is at work). I can recall when I was a graduate student it seemed there was a veritable industry around presenting concurrent GC algorithms in this very publication, with proofs of correctness. In fact, these proofs were offered because they were among the hardest correctness proofs researchers could conceive. Needless to say, this suggests the difficulty of getting concurrent GC algorithms right, much less translating them into correct implementations.

It is quite a tour de force that the authors of the following paper have built a provably correct real-time collector for reconfigurable hardware.

Concurrent GC alone is not enough to achieve real-time storage management. You also need provable bounds on the time to collect and the maximum memory needed by your program running under this scheme. Firstly, the collector cannot fall behind the mutator: it must be able to collect disused memory and recycle it for mutator use at least as fast as the mutator allocates memory units. An implication is that unless you can impose an artificial bound on the mutator's allocation rate, your collector must not only be concurrent but also *fast*.

While people have developed a wide range of GC algorithms, we are concerned here with ones that start from program variables (*roots*) and follow pointers from object to object, finding all the *reachable* objects. Such *tracing collectors* work in cycles: from the roots, trace the reachable objects, reclaim what is left, then go on to the next cycle.

Though it is a bit of a misnomer, GC developers also call reachable objects *live*, and their total volume at a given point in execution is the *live size*. In addition to a fast enough concurrent GC algorithm, for real-time GC you need not only a hard bound on the program's maximum live size—probably needed for real-time mutator behavior anyway—but also a bound on the amount of garbage (*unreachable* objects) that will accumulate during a collection cycle.

It is thus quite a tour de force that the authors of the following paper have built a provably correct real-time collector for reconfigurable hardware (field programmable gate arrays). How can this be? It turns out the FPGA setting makes the problem simpler in some ways than is the case with software GC running on stock processors. They can reasonably impose simple uniformity on the memory and its layout; they can exploit single-clock read-modify-write steps; and perhaps most importantly they have, via dual ported memory, completely concurrent memory access. This all leads to one of the cleanest and perhaps most understandable implementations of a concurrent GC that has ever been presented. The other prerequisites for real-time collection also follow easily. It is difficult to find the right word to express the feeling I get seeing such a sophisticated algorithmic idea reduced to such a straightforward hardware implementation—"Cool!" will have to suffice.

This work does not appear to offer a direct path to simple real-time GC support for software implementations on stock hardware. At the same time, it helps to inform that work through its contribution to knowledge about real-time GC beyond its benefits to practice. In particular, it shows how tight a bound we can achieve on the total space needed for a no-pause collector to run. I feel certain it will inspire creative approaches that will help bring garbage collection into acceptance in almost every corner of the system implementation space. C

Eliot Moss (moss@cs.umass.edu) is a professor in the Department of Computer Science at the University of Massachusetts, Amherst.

DOI:10.1145/2534706.2534726

And Then There Were None: A Stall-Free Real-Time Garbage Collector for Reconfigurable Hardware

By David F. Bacon, Perry Cheng, and Sunil Shukla

1. INTRODUCTION

The end of frequency scaling has driven architects and developers to parallelism in search of performance. However, general-purpose MIMD parallelism can be inefficient and power-hungry, with power rapidly becoming the limiting factor. This has led the search for performance to non-traditional chip architectures like GPUs and other more radical architectures. The most radical computing platform of all is reconfigurable hardware, in the form of Field-Programmable Gate Arrays (FPGAs).

FPGAs are now available with over one million programmable logic cells and 8MB of on-chip "Block RAM," providing a massive amount of bit-level parallelism combined with single-cycle access to on-chip memory. Furthermore, because that memory is distributed over the chip in distinct 36Kb units, the potential internal bandwidth is very high.

However, programming methodology for FPGAs has lagged behind their capacity, consequently reducing their applicability to general-purpose computing. The common languages in this domain are still hardware description languages (VHDL and Verilog) in which the only abstractions are bits, arrays of bits, registers, wires, and so on. The entire approach to programming them is oriented around the synthesis of a chip that happens to be reconfigurable, as opposed to programming a general-purpose device.

Recent research has focused on raising the level of abstraction and programmability to that of high-level software-based programming languages: the Scala-based Chisel framework,[4] C#-based Kiwi project[10] and the Liquid Metal project, which has developed the Lime language[3] based on Java. However, until now, whether programmers are writing in low-level HDLs or high-level languages like Chisel and Lime, use of dynamic memory management has only just begun to be explored,[8, 14] and use of garbage collection has been nonexistent.

In this article, we present a garbage collector synthesized directly to hardware, capable of collecting a heap of *uniform* objects completely concurrently. These uniform heaps (*miniheaps*) are composed entirely of objects of a fixed shape. Thus, the size of the data fields and the location of pointers of each object are fixed. We trade some flexibility in the memory layout for large gains in collector performance. In the FPGA domain, this trade-off makes sense: due to the distributed nature of the memory, it is common to build pipelined designs where each stage of the pipeline maintains its own internal data structures that are able to access their local block RAM in parallel with other pipeline stages. Furthermore, fixed data layouts can provide order-of-magnitude better performance because they allow designs which deterministically process one operation per clock cycle.

Algorithmically, our collector is a Yuasa-style snapshot-at-the-beginning collector,[16] with a linear sweep. By taking advantage of hardware structures like dual-ported memories, the ability to simultaneously read and write a register in a single cycle, and to atomically distribute a control signal across the entire system, we are able to develop a collector that *never* interferes with the mutator. Furthermore, the mutator has single-cycle access to memory. Arbitration circuits delay some collector operations by one cycle in favor of mutator operations, but the collector can keep up with a mutator even when it performs a memory operation every cycle (allocations are limited to one every other cycle).

Our stall-free collector can be used directly with programs hand-written in hardware description languages. Alternatively, it can be part of a hardware "runtime system" used by high-level language[3, 10] systems including dynamic memory allocation. For evaluation purposes, we have also implemented a stop-the-world variant and a malloc/free-style system. Using a very allocation-intensive application, the three collectors are compared in terms of memory usage, clock frequency, throughput (cycles and wall-clock time), and application stalls. We also present analytic closed-form worst-case bounds for the minimum heap size required for 0-stall real-time behavior, which are empirically validated. Further details, such as the energy consumption and additional benchmarking, are available in the original paper.[6]

2. FPGA BACKGROUND

FPGAs are programmable logic devices consisting of many discrete resources units that are enabled and connected based on the user application. Resources include look-up tables (LUTs) which can be used to implement combinational logic, and flip-flops which can be used to implement sequential logic or

The original version of this paper was published in *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'12)*, June 2012, ACM.

state. On the Xilinx FPGAs, which we use in this work, LUTs and flip-flops are grouped into *slices*, which is the standard unit in which resource consumption is reported for FPGAs.

Particularly important to this work are the discrete memory blocks, called Block RAMs (BRAMs), available on the FPGA. BRAMs are dual-ported specialized memory structures embedded within the FPGA for resource-efficient implementation of random- and sequential-access memories.

The Xilinx Virtex-5 LX330T[15] device that we use in this paper has a total BRAM capacity of 1.45MB. A single BRAM in a Virtex-5 FPGA can store up to 36Kb of memory. An important feature of BRAM is that it can be organized in various form factors (addressable range and data width combinations). For example, a 36Kb BRAM can be configured as $36K \times 1$ (36K 1-bit locations), $16K \times 2$, $8K \times 4$, $4K \times 9$, $2K \times 18$, or $1K \times 36$ memory.

A 36Kb BRAM can also be used as two independent 18Kb BRAMs. Larger logical memory structures can be built by cascading multiple BRAMs. Any logical memory structure in the design which is not a multiple of 18Kb would lead to quantization (or, in memory system parlance, "fragmentation").

The quantization effect can be considerable depending on the logical memory structure in the design (and is explored in Section 7). A BRAM can be used as a true dual ported (TDP) RAM providing two fully independent read-write ports. Furthermore, each port supports the following three modes for the read-write operation: read-before-write (previously stored data is available on the read bus), read-after-write (newly written data is available on the read bus), and no-change (read data bus output remains unchanged). Our collector makes significant use of read-before-write for features like the Yuasa-style write barrier.[16]

BRAMs can also be configured for use as FIFOs rather than as random access memories; we make use of this feature for implementing the mark queues in the tracing phase of the collector.

FPGAs are typically packaged on boards with dedicated off-chip DRAM and/or SRAM which can be accessed via a memory controller synthesized for the FPGA. Such memory could be used to implement much larger heap structures. However, we do not consider use of DRAM or SRAM in this paper because we are focusing on high-performance designs with highly deterministic (single cycle) behavior.

3. MEMORY ARCHITECTURE

The memory architecture—that is, the way in which object fields are laid out in memory, and the free list is maintained—is common to our support of both malloc/free and garbage-collected abstractions. In this section we describe our memory architecture as well as some of the alternatives, and discuss the trade-offs qualitatively. Some trade-offs are explored quantitatively in Section 7.

Since memory structures within an FPGA are typically and of necessity far more uniform than in a conventional software heap, we organize memory into one or more *miniheaps*, in which objects have a fixed size and "shape" in terms of division between pointer and data fields.

3.1. Miniheap interface

Each miniheap has an interface allowing objects allocation (and freeing when using explicit memory management), and operations for individual data fields to be read or written. In this article, we consider only miniheaps with one or two pointer fields and one or two data fields. This is sufficient for implementing stacks, lists, queues, and tree data structures. FPGA modules for common applications like packet processing, compression, etc. are covered by such structures.

Our design allows an arbitrary number of data fields. Increasing the number of pointer fields is straightforward for malloc-style memory. However, for garbage collected memory, the extension would require additional logic. We believe this is relatively straightforward to implement (and include details below) but the experimental results in this paper are confined to one- and two-pointer objects.

3.2. Miniheap with malloc/free

There are many ways in which the interface in Section 3.1 can be implemented. Fundamentally, these represent a time/space trade-off between the number of available parallel operations, and the amount of hardware resources consumed.

For FPGAs, one specifies a logical memory block with a desired data width and number of entries, and the synthesis tools attempt to allocate the required number of individual Block RAMs as efficiently as possible, using various packing strategies. We refer to the BRAMs for such a logical memory block as a *BRAM set*.

In our design we use one BRAM set for each field in the object. For example, if there are two pointer fields and one data field, then there are three BRAM sets.

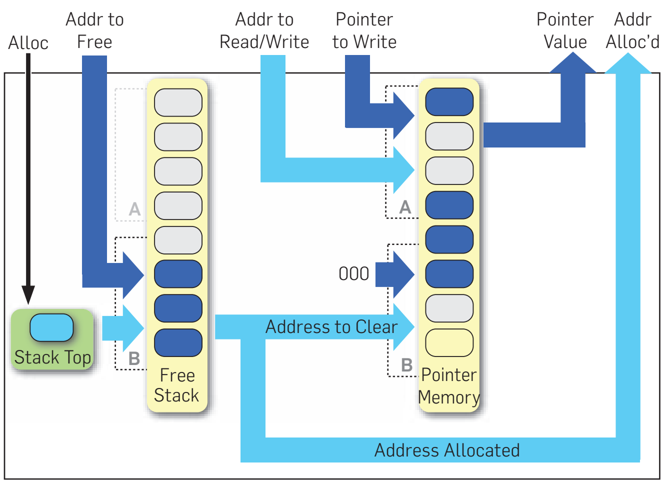
The non-pointer field has a natural width associated with its data type (for instance 32 bits). However, for a miniheap of size N, the pointer fields need only be $\lceil \log_2 N \rceil$ bits wide. Because data widths on the FPGA are completely customizable, we use precisely the required number of bits. Thus a larger miniheap will increase in size not only because of the number of entries, but because the pointer fields themselves become larger. As in software, the pointer value 0 is reserved to mean "`null`", so a miniheap of size N can really only store $N - 1$ objects.

A high-level block diagram of the memory manager is shown in Figure 1. It shows the primary data and control fields of the memory module, although many of the signals have been elided to simplify the diagram. For clarity of presentation it shows a single object field, of pointer type (Pointer Memory), which is stored in a single BRAM set. A second BRAM set (Free Stack) is used to store a stack of free objects. Using the read-before-write property of BRAMs we are able to implement the two functions (alloc and free) using port B of the BRAM set (Free Stack), leaving port A unused.

For an object with f fields, there are f BRAM sets with associated interfaces for the write and read values (but *not* an additional address port). There is only a single free stack, regardless of how many fields the object has.

The *Alloc* signal is a one-bit signal used to implement the `malloc` operation. A register is used to hold the value of the stack top. Assuming it is non-zero, it is decremented and then presented on port B of the Free Stack BRAM set, in *read* mode. The resulting pointer to a free field is then returned (*Addr Alloc'd*), but is also fed to port B of the Pointer Memory, in *write* mode with the write value

Figure 1. Memory module design for malloc/free interface, showing a single pointer field. BRAMs are in yellow, with the dual ports (A/B) shown. Ovals designate pointer fields; those in blue are in use.



hard-wired to 000 (or "`null`").

To free an object, the pointer is presented to the memory manager (*Addr to Free*). The Stack Top register is used as the address for the Free Stack BRAM set on port B, in write mode, with the data value *Addr to Free*. Then the Stack Top register is incremented. This causes the pointer to the freed object to be pushed onto the Free Stack.

In order to read or write a field in the Pointer Memory, the *Addr to Read/Write* is presented, and, if writing, a *Pointer to Write*. This uses port A of the BRAM set in either read or write mode, returning a value on the *Pointer Value* port in the former case.

Note that this design, by taking advantage of the dual-porting of the BRAMs, allows a read or write to proceed in parallel with an allocate or free.

Threaded free list. A common software optimization would be to represent the free objects not as a stack of pointers, but as a linked list threaded through the unused objects (that is, a linked list through the first pointer field). Since the set of allocated and free objects are mutually exclusive, this space optimization is essentially free modulo cache locality effects.

However, in hardware, this causes resource contention on the BRAM set containing the first pointer (since it is doing double duty). Thus parallelism is reduced: read or write operations on the first pointer cannot be performed in the same cycle as `malloc` or `free`, and the latter require two cycles rather than one. Our choice in using a stack is deliberate as the modest increased use of space is far less costly than the potential resource contention.

4. GARBAGE COLLECTOR DESIGN

We now describe the implementation of both a stop-the-world and a fully concurrent collector in hardware. In software, the architectures of these two styles of collector are quite different. In hardware, the gap is much smaller, as the same fundamental structures and interfaces are used.

The concurrent collector has a few extra data structures (implemented with BRAMs) and requires more careful allocation of BRAM ports to avoid contention, but these features do not negatively affect its use by the stop-the-world collector. Therefore, we will present the concurrent collector design, and merely mention here that the stop-the-world variant omits the shadow register(s) from the root engine, the write barrier register and logic from the trace engine, and the used map and logic from the sweep engine. Our design comprises three separate components, which handle the atomic root snapshot, tracing, and sweeping.

4.1. Background: Yuasa's snapshot algorithm

Before delving into the details of our implementation, we describe Yuasa's snapshot algorithm[16] which is the basis of our implementation. While the mechanics in hardware are quite different, it is interesting to note that implementing in hardware allows us to achieve a higher degree of concurrency and determinism than state-of-the-art software algorithms, but without having to incorporate more sophisticated algorithmic techniques developed in the interim.

The fundamental principle of the snapshot algorithm is that when collection is initiated, a *logical* snapshot of the heap is taken. The collector then runs in this logical snapshot, and collects everything that was garbage at snapshot time.

In Yuasa's original algorithm, the snapshot consisted of the registers, stacks, and global variables. This set of pointers was gathered synchronously (since then, much research has been devoted to avoiding the need for any global synchronization at snapshot time or during phase transitions[2]).

Once the roots have been gathered, the mutator is allowed to proceed and the collector runs concurrently, marking the transitive closure of the roots.

If the mutator concurrently modifies the heap, its only obligation is to make sure that the collector can still find all of the objects that existed in the heap at snapshot time. This is accomplished by the use of a *write barrier*: before any pointer is overwritten, it is recorded in a buffer and treated as a root for the purposes of collection.

Objects that are freshly allocated during a collection are not eligible for collection (they are "allocated black" in the parlance of collector literature).

The advantages of the snapshot algorithm are simplicity and determinism. Since it operates on a logical snapshot at an instant in time, the invariants of the algorithm are easy to describe. In addition, termination is simple and deterministic, since the amount of work is bounded at the instant that collection begins.

4.2. Root snapshot

The concurrent collector uses the snapshot-at-the-beginning algorithm described above. Yuasa's original algorithm required a global pause while the snapshot was taken by recording the roots; since then real-time collectors have endeavored to reduce the pause required by the root snapshot. In hardware, we are able to completely eliminate the snapshot pause by taking advantage of the parallelism and synchronization available in the hardware.

The snapshot must take two types of roots into account: those in registers, and those on the stack. Figure 2 shows the root snapshot module, simplified to show a single stack and a single register.

The snapshot is controlled by the *GC* input signal, which goes high for one clock cycle at the beginning of collection. The snapshot is defined as the state of the memory at the beginning of the *next cycle* after the *GC* signal goes high. This allows some setup time and reduces synchronization requirements. Note that if the *GC* signal is asserted while a collection is already under way, then the trigger is ignored.

The register snapshot is obtained by using a shadow register. In the cycle after the *GC* signal goes high, the value of the register is copied into the shadow register. This can happen even if the register is also written by the mutator in the same cycle, since the new value will not be latched until the end of the cycle.

The stack snapshot is obtained by having another register in addition to the Stack Top register, called the Scan Pointer. In the same cycle that the *GC* signal goes high, the value of the Stack Top pointer minus one is written into the Scan Pointer (because the Stack Top points to the entry above the actual top value). Beginning in the following cycle, the Scan Pointer is used as the source address to port B of the BRAM set containing the stack, and the pointer is read out, going through the MUX and emerging on the *Root to Add* port from the snapshot module. The Scan Pointer is also decremented in preparation for the following cycle.

Note that the mutator can continue to use the stack via port A of the BRAM set, while the snapshot uses port B. And since the mutator cannot pop values off the stack faster than the collector can read them out, the property is preserved that the snapshot contains *exactly* those roots that existed in the cycle following the *GC* signal.

A detail omitted from the diagram is that a state machine is required to sequence the values from the stack and the shadow register(s) through the MUX to the *Root to Add* port. Note that the values from the stack must be processed first, because the stack snapshot technique relies on staying ahead of the mutator without any explicit synchronization.

If multiple stacks were desired, then a "shadow" stack would be required to hold values as they were read out before the mutator could overwrite them, which could then be sequenced onto the *Root to Add* port.

As will be seen in Section 4.4, collection is triggered (only) by an allocation that causes free space to drop below a threshold. Therefore the generation of root snapshot logic only needs to consider those hardware states in which this might occur. Any register or stack not live in those states can be safely ignored.

4.3. Tracing

The tracing engine, along with a single pointer memory (corresponding to a single pointer field in an object) is shown in Figure 3. It provides the same mutator interface as the malloc/free style memory manager of Figure 1: *Addr to Read/Write*, *Pointer to Write*, and *Pointer Value*—except that the external interface *Addr to Free* is replaced by the internal interface (denoted in red) *Addr to Clear*, which is generated by the Sweep module (described in Section 4.4).

The only additional interface is the *Root to Add* port which takes its inputs from the output port of the same name of the Root Engine in Figure 2.

As it executes, there are three sources of pointers for the engine to trace: externally added roots from the snapshot, internally traced roots from the pointer memory, and overwritten pointers from the pointer memory (captured with a Yuasa-style barrier to maintain the snapshot property). The different pointer sources flow through a MUX, and on each cycle a pointer can be presented to the Mark Map, which contains one bit for each of the *N* memory locations.

Using the BRAM read-before-write mode, the old mark value is read, and then the mark value is unconditionally set to 1. If the old mark value is 0, this pointer has not yet been traversed, so the negation of the old mark value (indicated by the bubble) is used to control whether the pointer is added to the Mark Queue (note that this means that all values in the Mark Queue have been filtered, so at most $N - 1$ values can flow through the queue).

Pointers from the Mark Queue are presented as a read address on port B of the Pointer Memory, and if the fetched

Figure 2. Atomic root snapshot engine.

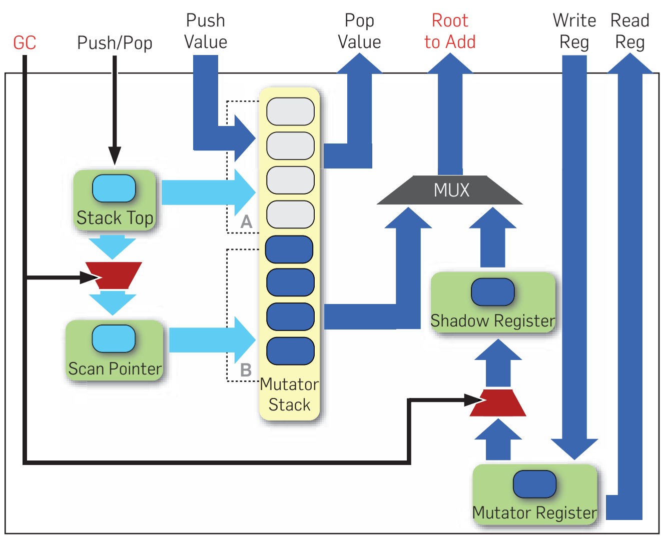
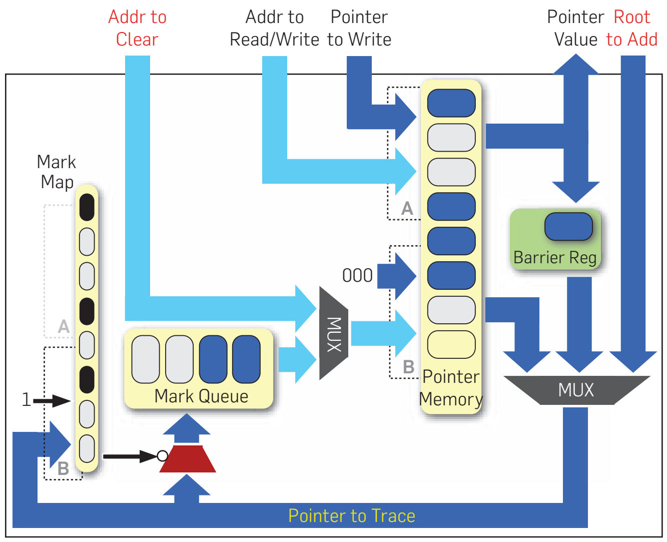


Figure 3. Tracing engine and one-pointer memory.



values are non-null, they are fed through the MUX and thence to the marking step.

The write barrier is implemented by using port A of the Pointer Memory BRAM in read-before-write mode. When the mutator writes a pointer, the old value is read out first and placed into the Barrier Reg. This is subsequently fed through the MUX and marked (the timing and arbitration is discussed below).

Given the three BRAMs involved in the marking process, processing one pointer requires 3 cycles. However, the marking engine is implemented as a *3-stage pipeline*, so it is able to sustain a throughput of one pointer per cycle.

Trace engine pairing. For objects with two pointers, two trace engines are paired together to maximize resource usage (this is not shown in the figure). Since each trace engine only uses one port of the mark map, both engines can mark concurrently.

The next item to mark is always taken from the longer queue. When there is only one item to enqueue, it is placed on the shorter queue. Using this design, we provision each of the 2 queues to be of size $3\ N/8 + R$ (where R is the maximum number of roots), which guarantees that the queues will never overflow.

On each cycle, one pointer is removed from the queues, and the two pointers in the object retrieved are examined and potentially marked and enqueued.

To minimize interference due to write barrier, our design has two write barrier registers. The write barrier values are not processed until a pair is formed and coupled with the fact that there are two mark queues, the mark engines can make progress every other cycle even if the application is performing one write per cycle.

Trace termination and WCET effects. The termination protocol for marking is simple: once the last item from the mark queues is popped (both mark queues become empty), it takes 2 or 3 cycles for the trace engine to finish the current pipeline. If the two pointers returned by the heap are null, then the mark process is terminated in the second cycle as there is no need to read the mark bits in this case. Otherwise the mark bit for the non-null pointers are read to ensure that both pointers are marked, in which case the mark phase is terminated in the third cycle.

Write barrier values arriving after the first cycle of termination can be ignored, since by the snapshot property they would either have to be newly allocated or else discovered by tracing the heap.

However, note that some (realistic) data structures, in particular linked lists, will cause a pathological behavior, in which a pointer is marked, removed from the queue, which will appear empty, and then 2 cycles later, the next pointer from the linked list will be enqueued. So while the pipeline can sustain marking one object per cycle, pipeline bubbles will occur which reduce that throughput.

4.4. Sweeping

Once tracing is complete, the sweep phase begins, in which memory is reclaimed. The high-level design is shown in Figure 4. The sweep engine also handles allocation requests and maintains the stack of pointers to free memory (Free Stack). The Mark Map here is the same Mark Map as in Figure 3.

When an *Alloc* request arrives from the mutator, the Stack Top register is used to remove a pointer to a free object from the Free Stack, and the stack pointer is decremented. If the stack pointer falls below a certain level (we typically use 25%), then a garbage collection is triggered by raising the *GC* signal which is connected to the root snapshot engine (Figure 2).

The address popped from the Free Stack is returned to the mutator on the *Addr Alloc'd* port. It is also used to set the object's entry in the Used Map, to `01`, meaning "freshly allocated" (and thus "black"). A value of `00` means "free", in which case the object is on the Free Stack.

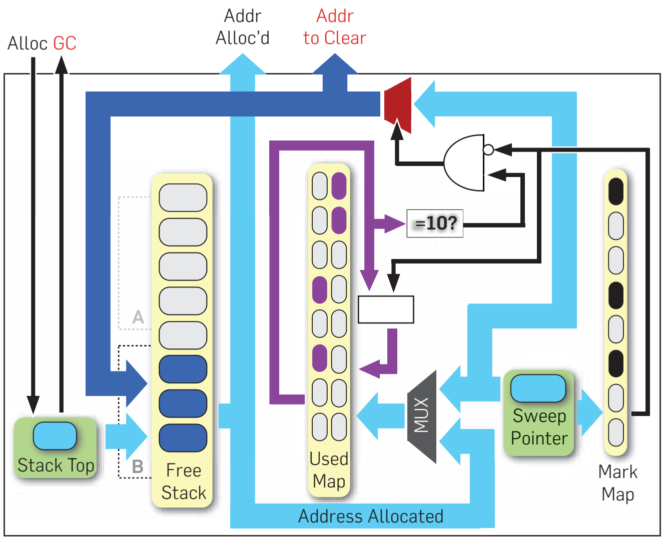
When tracing is completed, sweeping begins in the next cycle. Sweeping is a simple linear scan. The Sweep Pointer is initialized to 1 (since slot 0 is `null`), and on every cycle (except when preempted by allocation) the sweep pointer is presented to both the Mark Map and the Used Map.

If an object is marked, its Used Map entry is set to `10`. If an object is not marked and its used map entry is `10` (the *and* gate in the figure) then the used map entry is reset to `00`. Although only 3 states are used, the particular choice of bit pattern avoids unneeded logic in the critical path. The resulting signal is also used to control whether the current Sweep Pointer address is going to be freed. If so, it is pushed onto the Free Stack and also output on the *Addr to Clear* port, which is connected to the mark engine so that the data values being freed are zeroed out.

Note that since clearing only occurs during sweeping, there is no contention for the Pointer Memory port in the trace engine between clearing and marking. Furthermore, an allocation and a free may happen in the same cycle: the top-of-stack is accessed using read-before-write mode and returned as the *Addr Alloc'd*, and then the newly freed object is pushed back.

When an object is allocated, its entry in the Mark Map is *not* set (otherwise an extra interlock would be required). This means that the tracing engine may encounter newly allocated objects in its marking pipeline (via newly installed pointers in the heap), albeit at most once since they will then be marked. This also affects WCET analysis, as we will see in the next section.

Figure 4. Free stack and sweeping engine.



5. REAL-TIME BEHAVIOR

First of all, we note that since the design of our real-time collector allows mutation and collection to occur unconditionally together in a single cycle, the minimum mutator utilization (or MMU[7]), is 100% unless insufficient resources are dedicated to the heap.

Furthermore, unlike software-based collectors,[5, 11] the system is fully deterministic because we can analyze the worst case behavior down to the (machine) cycle.

Given R is the maximum number of roots, N is the size of the heap, then the worst-case time (in cycles) for garbage collection is

$$T = T_R + T_M + T_W + T_X + T_S + T_A \tag{1}$$

where T_R is the time to snapshot the roots, T_M is the time (in cycles) to mark, T_S is the time to sweep, and T_W is the time lost to write barriers during marking, T_X is the time lost to blackening newly allocated objects during marking, and T_A is time lost to allocations during sweeping.

In the worst case, without any knowledge of the application,

$$T_R = R+2 \quad T_M = 3N+3 \quad T_W = 0 \quad T_X = 0 \quad T_S = N$$

The reasoning for these quantities follows. During the snapshot phase, we can place one root into the mark queue every cycle, plus one cycle to start and finish the phase, accounting for $R + 2$. During marking, there could be N objects in the heap, configured as a linked list which caused the mark pipeline to stall for two cycles on each object, plus 3 cycles to terminate. Sweeping is unaffected by application characteristics, and always takes N cycles. Preemption of the collector by mutator write barriers (T_W) does not factor into the worst-case analysis because the write barrier work is overlapped with the collector stalls. Extra mark operations to blacken newly allocated objects (T_X) also simply fill stall cycles.

Our design allows an allocation operation *in every cycle*, but allocation preempts the sweep phase, meaning that such an allocation rate can only be sustained in short bursts. The largest sustainable allocation rate is 0.5—otherwise the heap would be exhausted before sweeping completed. Thus $T_A = N$ and

$$T_{\text{worst}} = R + 5N + 5 \tag{2}$$

5.1. Application-specific analysis

Real-time analysis typically takes advantage of at least some application-specific knowledge. This is likely to be particularly true of hardware-based systems. Fortunately, the structure of such systems makes it more likely that such factors can be quantified to a high degree of precision, for example by looking at operations per clock cycle in the synthesized design.

Let μ be the average number of mutations per cycle ($\mu \leq 1$), α be the average number of allocations per cycle ($\alpha < 0.5$), and m be the maximum number of live data objects in the heap at any one time ($m < N$). Then we can more precisely estimate

$$T'_M = 3m+3 \quad T'_X = \alpha T'_M \quad T'_W = \frac{\mu}{2-\mu}m \quad T'_A = \frac{\alpha}{1-\alpha}N$$

Note that both α and μ can only be averaged over a *time window* guaranteed to be less than or equal to the phases which they influence; m is a safe window size.

The largest inaccuracy is still due to pipeline stalls during marking, for which worst- and average-case behavior can be very different. We therefore let B be the number of pipeline stalls ($0 \leq B \leq 2m$), so an even more precise bound on marking is $T''_M = m+B+3$ (and also improving $T''_X = \alpha\, T''_M$).

For a linked list, $B = 2m$; for three linked lists each with its own root, $B = 0$. We hypothesize that for the heap considered as a forest without back-edges, B is bounded by the number of levels of width 1 plus the number of levels of width 2 (when the width is 3 or greater, there is enough parallelism to keep the 3-stage pipeline full and avoid stalls).

Using these application-specific estimates, we then are able to bound the worst-case execution time (WCET) of collection as

$$T_{\max} = \left(\frac{1}{1-\alpha}\right)\left(R+B+5+\frac{2}{2-\mu}m+\frac{N}{1-\alpha}\right) \tag{3}$$

5.2. Minimum heap size

Once the worst-case execution time for collection is known, we can solve for the minimum heap size in which the collector can run with real-time behavior (zero stalls). Note that if a heap size is deliberately chosen below this minimum, the allocator may experience an out-of-memory condition. As a starting point, m objects must be available for the live data. While a collection taking time $T_{\max}$ takes place, another $\alpha T_{\max}$ objects can be allocated. However, there may also be $\alpha T_{\max}$ floating garbage from the previous cycle when a collection starts. Thus the minimum heap size is

$$N_{\min} = m + 2\alpha\, T_{\max} \tag{4}$$

and if we denote the non-size-dependent portion of $T_{\max}$ from equation (3) by

$$K = \left(\frac{1}{1-\alpha}\right)\left(R+B+5+\frac{2}{2-\mu}m\right)$$

then we can solve for

$$\begin{aligned} N_{\min} &= m+2\alpha T_{\max} \\ &= m+2\alpha\left(K+\frac{N_{\min}}{(1-\alpha)^2}\right) \\ N_{\min} &= \frac{(1-\alpha)^2(m+2\alpha K)}{1-4\alpha+\alpha^2} \end{aligned} \tag{5}$$

6. EXPERIMENTAL METHODOLOGY

Since we have implemented the first collector of this kind, we cannot leverage a preexisting set of benchmarks for evaluation. Here, we present the results of a very allocation-intensive microbenchmark, a doubly-ended queue (deque), which commonly occurs in many applications. In our HDL implementation, the doubly-linked list can be modified by pushes and pops to either the front or back. The workload consists of a pseudorandom sequence of such operations while keeping the maximum amount of live data close to but no more than 8192. Because there is almost no computation performed on the list elements, this benchmark is very allocation-intensive. Moreover, the linear nature of the data structure prevents the collector from taking advantage of graph fanout. These two factors combine to present a great challenge to the collector.

It is important to note that because of the very high degree of determinism in the hardware, and in our collector implementation, such micro-benchmarks can provide a far more accurate picture of performance than in typical evaluations of CPU-based collectors running in software. There are no cache effects, no time-slicing, and no interrupts. Because these higher order effects are absent, the performance behavior presented to the mutator by the collector and vice versa is completely captured by the memory management API at a cycle-accurate level. We validate this experimentally by showing that the estimates for collection time and minimum real-time heap size (from Section 5.1) are highly accurate.

A given micro-benchmark can be paired with one of the three memory management implementations (Malloc, stop-the-world GC, and real-time GC). Furthermore, these are parameterized by the size of the miniheap, and for the collectors, the trigger at which to start collection (although for most purposes, we simply trigger when free space falls below 25%). We call these *design points*.

Our experiments are performed by using the Xilinx ISE 13.4 synthesis tools on a Xilinx Virtex-5 LX330T,[15] which is the largest FPGA within the Virtex5 LXT product line. The LX330T has 51,840 slices and 11,664Kb (1.45MB) of Block RAM. Fabricated in 65 nm technology, the chip is theoretically capable of being clocked at up to 550 MHz, but realistic designs generally run between 100 and 300 MHz.

7. EVALUATION

We begin by examining the cost, in terms of static resources, of the 3 memory managers—malloc/free ("Malloc"), stop-the-world collection ("STW"), and real-time concurrent collection ("RTGC"). For these purposes we synthesize the memory manager in the absence of any application. This provides insight into the cost of the memory management itself, and also provides an upper bound on the performance of actual applications (since they can only use more resources or cause the clock frequency to decline).

We evaluate design points at heap sizes (in objects) from 1K to 64K in powers of 2. For these purposes we use an object layout of two pointers and one 32-bit data field. For brevity, we omit detailed usage of non-BRAM logic resources. It is enough to note that for all cases, the logic consumption is under 1% for all 3 variants and at all heaps sizes.

Figure 5 shows BRAM consumption. Because we have chosen powers of 2 for heap sizes, the largest heap size only uses 60% of the BRAM resources (one is of course free to choose other sizes). At the smaller heap sizes, garbage collectors consume up to 80% more BRAMs than Malloc. However, at realistic heap sizes, the figure drops to 24%. In addition, RTGC requires about 2–12% more memory than STW since it requires the additional 2-bit wide Used Map to cope with concurrent allocation. Fragmentation is noticeable but not a major factor, ranging from 11–31% for Malloc and 11–53% for garbage collection. As before, at larger heap sizes, the fragmentation decreases. Some wastage can be avoided by choosing heap sizes more carefully, not necessarily a power of 2, by noting that BRAMs are available in 18Kb blocks. However, some fragmentation loss is inherent in the quantization of BRAMs as they are chained together to form larger memories.

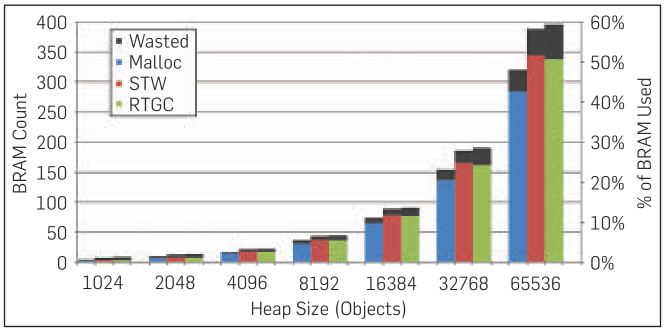
Finally, Figure 6 shows the synthesized clock frequency at different design points. Here we see a significant effect from the more complex logic for garbage collection: even though it consumes relatively little area, clock frequency for garbage collection is noticeably slower (15–39%) than Malloc across all design points. On the other hand, the difference between STW and RTGC is small with RTGC often faster. Regardless of the form of memory management, clock frequency declines as the heap becomes larger. However, the overall clock rate may very well be constrained by the application logic rather than the collector logic, as we will see below.

7.1. Throughput

So far we have discussed the costs of memory management in the absence of applications; we now consider what happens when the memory manager is "linked" to the Deque microbenchmark. Unlike the previous section, where we concentrated on the effects of a wide range of memory sizes on static chip resources, here we focus on a smaller range of sizes using a trace with a single maximum live data set of m = 8192 as described previously. We then vary the heap size N from m to $2m$ at fractional increments of $m/10$. As we make memory scarce, the resolution is also increased to $m/100$ to show how the system behaves at very tight conditions. Each design point requires a full synthesis of the hardware design which can affect the frequency, power, and execution time.

Figure 7 shows the throughput of the benchmark as the heap size varies for all 3 schemes. To understand the interaction of various effects, we not only examine the throughput both in cycle duration but also, since the synthesizable clock frequencies vary, in physical time. The Deque benchmark shows a different behavior. With much higher allocation and mutation rates ($\alpha = 0.07$ and $\mu = 0.13$), it is much more sensitive to collector activity. As seen in Figure 7(b), even at heap size $N = 2m$, STW consumes noticeably more cycles, rising to almost double the cycles

Figure 5. Block RAM usage, including Fragmentation.



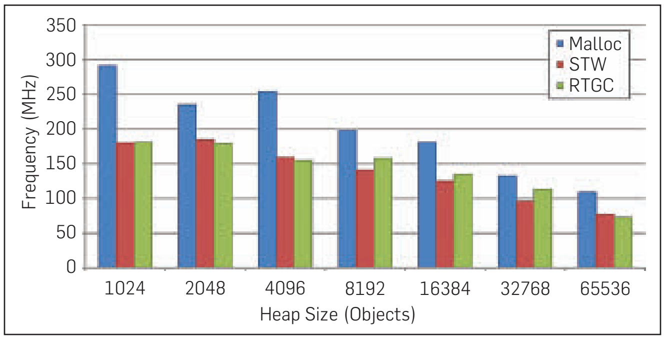
at $N = 1.1m$. By contrast RTGC consumes slightly fewer cycles than Malloc until it begins to experience stall cycles (non-real-time behavior) at $N = 1.4m$ because it falls behind the mutator.

The Deque benchmark has a very simple logic so any limitations on frequency introduced by the collector is magnified. The effect is seen clearly in Figure 7(b): Malloc synthesizes at a higher frequency, allowing it to make up RTGC's slight advantage in cycles and consume 25% less time on an average. STW suffers even more from the combined effect of a lower clock frequency and additional cycles due to synchronous collection. On average, RTGC is faster than STW by 14% and never interrupts the application.

These measurements reveal some surprising trends that are completely contrary to the expected trade-offs for software collectors: RTGC is actually *faster, more deterministic, and requires less heap space* than STW! There seems to be no reason to use STW because the natural advantage of implementing concurrency in hardware completely supersedes the traditional latency versus bandwidth trade-off.

Furthermore, RTGC allows applications to run at far lower multiples of the maximum live set m than possible for either real-time or stop-the-world collectors in software. RTGC is also only moderately slower than Malloc, meaning that the cost of abstraction is quite palatable. We do not show the results for other applications but note that, as predicted, this performance gap decreases as the application becomes more complex.

Figure 6. Synthesized clock frequency.



7.2. Validation of real-time bounds

Because the design and analysis of our concurrent collector is intended to be cycle-accurate, we can validate the time and space bounds of the collector with expectation that they will be fully met. Figure 8 shows the actual time spent in garbage collection and the analytic upper bound (T_{max} from equation 3). Note that we show both average as well as the maximum time spent in garbage collections. The heap size is chosen to be much tighter than in earlier graphs as our focus here is how the collector behaves when it is under stress (near N_{min} from equation 4). For convenience, we express heap size as a fraction (N/m) of the maximum amount of live data since our bounds are almost linear when considered in terms of m and N.

Average time spent in collection is always less than the predicted worst case with an actual difference of about 10% for both programs. We also show the amount of stalls experienced by the benchmark as a fraction of total time. At larger heap sizes, there are no stalls. As the heap size is reduced, there will come a point when the collector cannot keep up and the mutator's allocation request will fail. For the allocation-intensive Deque benchmarks, the failure point occurs at $1.43 \times m$. Our predicted N_{min} value of 1.457 is correctly above the actual failure points.

Because the average collection time includes multiple phases of a program, it can be significantly lower than the maximum collection time. We see that the gap between T_{max} and collection time shrinks from 10% to about 2% and 6% when one considers maximum rather than average collection time. For space, N_{min} has only a worst-case flavor as there is adequate heap space only if the heap is sufficient at every collection. The space bound is within 3% of when stalls begin. Our time and space bounds are not only empirically validated but are tight.

In general, time spent for a single collection falls as the heap size is decreased since the sweep phase will take less time. It may seem surprising that this happens even when the heap size is taken below N_{min}. However, falling below this safe point causes mutator stalls but does not penalize the collector at all. In fact, because the mutator is stalled, it can no longer interfere with the collector which will additionally, though very slightly, speed up collection. Of course, since the overall goal is to avoid mutator stalls, operating in this regime is inadvisable.

Figure 7. Throughput measurements for Deque. (a) Execution duration in cycles of Deque; (b) execution time in milliseconds of Deque.

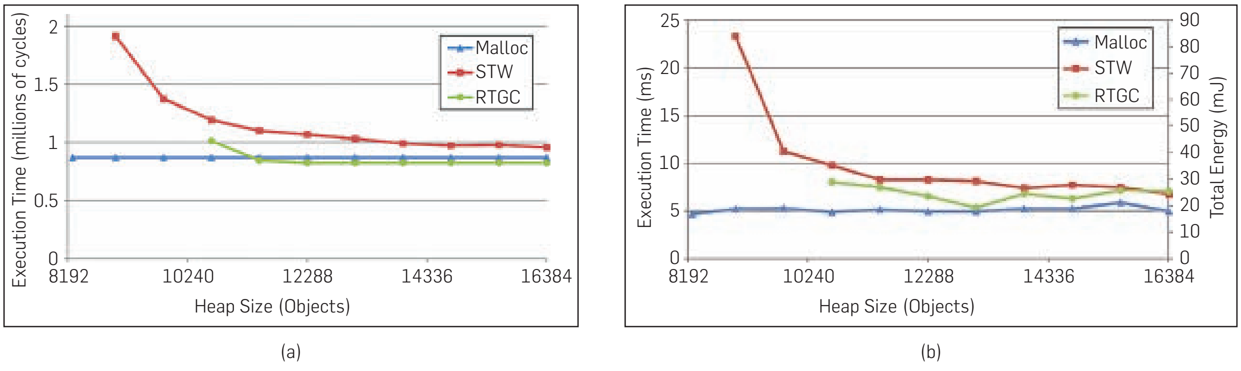
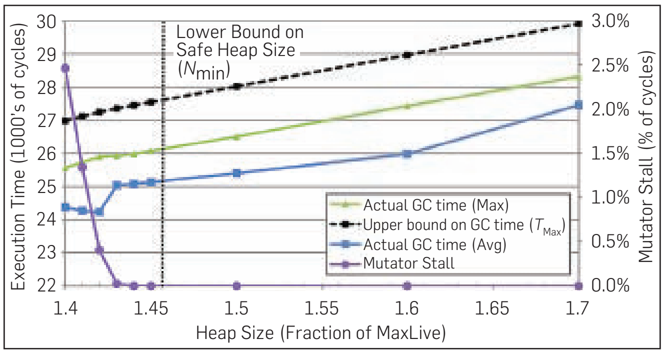


Figure 8. Comparison of predicted to actual duration and space usage.



8. RELATED WORK

We provide only a brief summary of related work and refer the reader to our full paper[6] for more details and citations.

There has been very little work on supporting high-level memory abstractions in reconfigurable hardware, and none on garbage collection. Simsa and Singh[14] have explored compilation of C subprograms that use malloc/free into VHDL or Bluespec. LEAP scratchpads[1] provide an expandable memory abstraction which presents a BRAM interface, but uses off-chip RAM if the structure is too large to fit, and transparently uses the on-chip BRAM as a cache. Such a system could be coupled with ours in order to provide a larger, virtualized memory, albeit at the expense of determinism and throughput.

Meyer[12] has built a special-purpose processor and an associated garbage collection co-processor on an Altera APEX FPGA. However, the design and the goals were very different. Meyer's collector is for a general-purpose heap allocated in DRAM, and for a program operating on what is for the most part a conventional CPU. The collector is implemented with a microcoded co-processor, and the general CPU is modified with a special pointer-related support. Because of this software-in-hardware approach, pauses can reach 500 cycles. In contrast, our approach is a fully custom logic that is much more tightly integrated with the memory, for "programs" that are also synthesized into hardware, and with deterministic single-cycle memory access. This allows us to attain zero pauses.

There are also garbage-collected systems[9, 13] in which the FPGA participates in a garbage collected heap but performs no collection itself. Yet others have designed special-purpose ASICs that perform a custom collection algorithm or provide a set of instructions to accelerate a more general class of algorithms. These are fundamentally different because the heap is on the CPU and not the FPGA.

9. CONCLUSION

We have described our design, implementation, and evaluation of the first garbage collectors to be completely synthesized into hardware. The real-time version causes *zero* cycles of interference with the mutator.

Careful implementation allows a closed-form analytic solution for worst-case execution time (WCET) of the collector, and a lower bound on heap size to achieve real-time behavior. These bounds are also cycle-accurate.

In software there are large trade-offs between stop-the-world and real-time collection in terms of throughput, latency, and space. Our measurements show that in hardware the real-time collector is faster, has lower (zero) latency, and can run effectively in less space.

This performance and determinism is not without cost: our collector only supports a single fixed object layout. Supporting larger objects with more pointers is a relatively straightforward extension of our design; supporting multiple object layouts is more challenging, but we believe can be achieved without sacrificing the fundamental advantages.

Garbage collection of programs synthesized to hardware is practical and realizable!

Acknowledgments

We thank the members of the Liquid Metal team at IBM Research who contributed insights, infrastructure, and a stimulating working environment: Joshua Auerbach, Ioana Baldini, Stephen Fink, and Rodric Rabbah. C

References

1. Adler, M. et al. Leap scratchpads: automatic memory and cache management for reconfigurable logic. In *Proceeding of International Symposium on Field Programmable Gate Arrays* (2011), 25–28.
2. Auerbach, J., Bacon, D.F., Cheng, P., Grove, D., Biron, B., Gracie, C., McCloskey, B., Micic, A., Sciampacone, R. Tax-and-spend: democratic scheduling for real-time garbage collection. In *Proceedings of the 8th ACM International Conference on Embedded Software* (2008), 245–254.
3. Auerbach, J., Bacon, D.F., Cheng, P., Rabbah, R. Lime: a Java-compatible and synthesizable language for heterogeneous architectures. In *Proceedings of ACM International Conference on Object Oriented Programming Systems, Languages, and Applications* (Oct. 2010), 89–108.
4. Bachrach, J., Huy Vo, B.R., Lee, Y., Waterman, A., Avidienis, R., Wawrzynek, J., Asanovic, K. Chisel: Constructing hardware in a scala embedded language. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC)* (Jun. 2012), 1212–1221.
5. Bacon, D.F., Cheng, P., Rajan, V.T. A real-time garbage collector with low overhead and consistent utilization. In *Proceedings of Symposium on Principles of Programming Languages* (New Orleans, Louisiana, Jan. 2003), 285–298.
6. Bacon, D.F., Cheng, P., Shukla, S. And then there were none: a stall-free real-time garbage collector for reconfigurable hardware. In *Proceedings of SIGPLAN Conference on Programming Language Design and Implementation* (Jun. 2012), 23–34.
7. Blelloch, G.E., Cheng, P. On bounding time and space for multiprocessor garbage collection. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, (Atlanta, Georgia, Jun. 1999), 104–117.
8. Cook, B. et al. Finding heap-bounds for hardware synthesis. In *Formal Methods in Computer-Aided Design* (Nov. 2009), 205–212.
9. Faes, P., Christiaens, M., Buytaert, D., Stroobandt, D. FPGA-aware garbage collection in Java. In *proceedings of the IEEE International Conference on Field Programmable Logic and Applications* (FPL 2005), 675–680.
10. Greaves, D., Singh S.. Kiwi: Synthesis of FPGA circuits from parallel programs. In *IEEE Symposium on Field-Programmable Custom Computing Machines* (2008).
11. Henriksson, R. *Scheduling Garbage Collection in Embedded Systems*. PhD thesis, Lund Institute of Technology (July 1998).
12. Meyer, M. An on-chip garbage collection coprocessor for embedded real-time systems. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time ComputingSystems and Applications* (2005), 517–524.
13. Schmidt, W.J., Nilsen, K.D. Performance of a hardware-assisted real-time garbage collector. In *Proceedings of the 6th International Conference on Architectural Support for Programming Languages andOperating Systems* (1994), 76–85.
14. Simsa, J., Singh, S. Designing hardware with dynamic memory abstraction. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (2010), 69–72.
15. Xilinx. Virtex-5 family overview. Technical Report DS100, Feb. 2009.
16. Yuasa, T. Real-time garbage collection on general-purpose machines. *J. Syst. Software 11*, 3 (Mar 1990), 181–198.

David F. Bacon, Perry Cheng, Sunil Shukla ([bacon, perry, skshukla]@us.ibm.com), IBM T.J. Watson Research Center, NY.

Distinguished Speakers Program

talks by and with technology leaders and innovators

Chapters • Colleges and Universities • Corporations • Agencies • Event Planners

A great speaker can make the difference between a good event and a WOW event!

The Association for Computing Machinery (ACM), the world's largest educational and scientific computing society, now provides colleges and universities, corporations, event and conference planners, and agencies – in addition to ACM local Chapters – with direct access to top technology leaders and innovators from nearly every sector of the computing industry.

Book the speaker for your next event through the ACM Distinguished Speakers Program (DSP) and deliver compelling and insightful content to your audience. **ACM will cover the cost of transporation for the speaker to travel to your event.** Our program features renowned thought leaders in academia, industry and government speaking about the most important topics in the computing and IT world today. Our booking process is simple and convenient. Please visit us at: **www.dsp.acm.org**. If you have questions, please send them to **acmdsp@acm.org**.

The ACM Distinguished Speakers Program is an excellent solution for:

Corporations Educate your technical staff, ramp up the knowledge of your team, and give your employees the opportunity to have their questions answered by experts in their field.

Colleges and Universities Expand the knowledge base of your students with exciting lectures and the chance to engage with a computing professional in their desired field of expertise.

Event and Conference Planners Use the ACM DSP to help find compelling speakers for your next conference and reduce your costs in the process.

ACM Local Chapters Boost attendance at your meetings with live talks by DSP speakers and keep your chapter members informed of the latest industry findings.

Captivating Speakers from Exceptional Companies, Colleges and Universities

DSP speakers represent a broad range of companies, colleges and universities, including:

IBM	Sony Pictures	Georgia Tech	University of British Columbia
Microsoft	McGill University	Carnegie Mellon University	Siemens Information Systems Bangalore
BBN Technologies	Tsinghua University	Stanford University	Lawrence Livermore National Laboratory
Raytheon	UCLA	University of Pennsylvania	National Institute of Standards and Technology

Topics for Every Interest

Over 400 lectures are available from 120 different speakers with topics covering:

Software	Web Topics	Career-Related Topics	Computer Graphics, Visualization and Interactive Techniques
Cloud and Delivery Methods	Computer Systems	Science and Computing	High Performance Computing
Emerging Technologies	Open Source	Artificial Intelligence	Human Computer Interaction
Engineering	Game Development	Mobile Computing	

Exceptional Quality Is Our Standard

The same ACM you know from our world-class Digital Library, magazines and journals is now putting the affordable and flexible Distinguished Speaker Program within reach of the computing community.

Microsoft® Research
The DSP is sponsored in part by Microsoft Europe

Association for Computing Machinery
Advancing Computing as a Science & Profession

Baylor University
Department of Computer Science
Lecturer of Computer Science

The Department of Computer Science seeks a dedicated teacher and program advocate for a lecturer position beginning August, 2014. The ideal candidate will have a master's degree or Ph.D. in Computer Science or a related area, a commitment to undergraduate education, effective communication and organization skills, and industry/academic experience in game development, especially with graphics and/or engine development. For position details and application information please visit: http://www.baylor.edu/hr/index.php?id=81302

Baylor, the world's largest Baptist university, holds a Carnegie classification as a "high-research" institution. Baylor's mission is to educate men and women for worldwide leadership and service by integrating academic excellence and Christian commitment within a caring community. Baylor is actively recruiting new faculty with a strong commitment to the classroom and an equally strong commitment to discovering new knowledge as Baylor aspires to become a top tier research university while reaffirming and deepening its distinctive Christian mission as described in Pro Futuris (http://www.baylor.edu/profuturis/).

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an AA/EEO employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Baylor University
Department of Computer Science
Assistant, Associate or Full Professor of Computer Science

The Department of Computer Science seeks a productive scholar and dedicated teacher for a tenured or tenure-track position beginning August, 2014. The ideal candidate will hold a terminal degree in Computer Science or a closely related field, demonstrate scholarly capability and an established and active independent research agenda in one of several core areas of interest, including, but not limited to, game design and development, software engineering, computational biology, informatics, machine learning and large-scale data mining. A successful candidate will also exhibit a passion for teaching and mentoring at the graduate and undergraduate level. For position details and application information please visit: http://www.baylor.edu/hr/index.php?id=81302

Baylor, the world's largest Baptist university, holds a Carnegie classification as a "high-research" institution. Baylor's mission is to educate men and women for worldwide leadership and service by integrating academic excellence and Christian commitment within a caring community. Baylor is actively recruiting new faculty with a strong commitment to the classroom and an equally strong commitment to discovering new knowledge as Baylor aspires to become a top tier research university while reaffirming and deepening its distinctive Christian mission as described in Pro Futuris (http://www.baylor.edu/profuturis/).

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an AA/EEO employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Boise State University
Department of Computer Science
Three Tenure/Tenure-Track Open-Rank Positions

The **Department of Computer Science at Boise State Universit**y invites applications for **three tenure/tenure-track open-rank positions**. Applicants should have a commitment to excellence in teaching and a desire to make significant contributions in research by collaborating with faculty and local industry to develop and sustain funded research programs. Senior applicants should have an established track record of research, teaching, and external funding. Preferences will be given to applicants working in the areas of Databases with an emphasis on Big Data, or Human-Computer Interaction with a particular emphasis on usability of user interfaces, or Visualization. An earned Ph.D. in Computer Science or a closely related field is required at the time of appointment.

Boise State has made a significant investment in the growth of the Computer Science department, which is a critical part of the vibrant software and high-tech industry in the Boise metropolitan area. New faculty lines, graduate student support, and a tutoring center have been added to the department. The department is committed to offering a high quality educational experience and in building its research capabilities. For more information, including details on how to apply, please visit us online at http://coen.boisestate.edu/cs/jobs.

Boise State University is strongly committed to achieving excellence through cultural diversity. The University actively encourages applications and nominations of women, persons of color, and members of other underrepresented groups. EEO/AA Institution, Veterans preference may be applicable.

Boston College
Computer Science Department
Assistant Professor, Computer Science

The Computer Science Department of Boston College invites applications for a tenure-track Assistant Professorship beginning September, 2014. Applications from all areas of Computer Science will be considered. Applicants should have a Ph.D. in Computer Science or related discipline, a strong research record, and a commitment to undergraduate teaching.

We will begin reviewing applications on December 1, 2013, and will continue considering applications until the position is filled. Additional information about the department and the position is available at www.cs.bc.edu. **Submit applications online at apply.interfolio.com/22805.**

Boston University
Department of Electrical & Computer Engineering
Faculty Search

The **Department of Electrical & Computer Engineering (ECE)** at **Boston University** (BU) is seeking candidates for anticipated faculty positions in Computer Engineering, with particular interest in tenure-track candidates in software, systems and cybersecurity. The Department is seeking to foster growth in the broad, interdisciplinary topics of energy, health, information systems, and cyberphysical systems. Candidates with research interests that transcend the traditional boundaries of ECE are strongly encouraged to apply. Joint appointments with other BU departments and with the Division of Systems Engineering are likely for candidates with appropriate experience and interests.

Qualified candidates must possess a relevant, earned PhD, and have a demonstrable ability to teach effectively at graduate and undergraduate levels, develop funded research programs in their area of expertise, and contribute to the tradition of excellence in research that is characteristic of the ECE Department. Self-motivated individuals who thrive on challenge and are eager to utilize their expertise to strengthen an ambitious program of departmental enhancement are desired. Women, minorities, and candidates from other underrepresented groups are especially encouraged to apply and help us continue building an exceptional 21st century university department.

ECE at BU is a world-class department with excellent resources that is steadily gaining national and international prominence for its exceptional research and education record. ECE is part of BU's rapidly growing and innovative College of Engineering, and currently consists of 40 faculty members, 200 graduate students, and 250 BS majors. Outstanding collaboration opportunities are available with nationally recognized medical centers and universities/colleges, nearby research centers, and industry throughout the Boston area.

Beyond its research and academic activities, BU has a lively, urban campus situated along the banks of the Charles River in Boston's historic Fenway-Kenmore neighborhood. The campus and surrounding areas offer limitless opportunities for recreational activities, from world-class art and performances to sporting events and fine dining.

Please visit http://www.bu.edu/ece/facultysearch for instructions on how to apply. Application deadline is December 31, 2013. The review of applications will begin on October 1, 2013. Therefore, applicants are encouraged to apply early. Boston University is an Equal Opportunity/Affirmative Action Employer.

California State University, Fullerton
Department of Computer Science
Assistant Professor

The Department of Computer Science invites applications for **three tenure-track** positions at the **Assistant Professor** level starting fall 2014. For a complete description of the department, the position, desired specialization and other qualifications, please visit http://diversity.fullerton.edu/.

California State University, San Bernardino
School of Computer Science and Engineering
Assistant Professor

The **School of Computer Science and Engineering** invites applications for a tenure track position at the **Assistant Professor** level. Candidates must have a Ph.D. or an earned Doctorate in Computer Engineering or a closely related field. We are particularly interested in candidates with strengths in embedded systems or signal processing. Other areas of computer engineering will also be considered. The position is primarily to support the B.S. in Computer Engineering. In addition, the school offers the degrees B.S. in Computer Science, B.S. in Bioinformatics, B.A. in Computer Systems, and M.S. in Computer Science.

The candidate must display potential for excellence in teaching and scholarly work. The candidate is expected to supervise student research at both the undergraduate and graduate levels, and to actively participate in other types of academic student advising. The candidate will actively contribute to the School's curriculum development. . The candidate will serve the School, College and University, as well as the community and the profession.

Women and underrepresented minorities are strongly encouraged to apply. For more information about the School of Computer Science and Engineering, please visit http://cse.csusb.edu

SALARY: Dependent on qualifications and experience.

BENEFITS: Generous medical, dental, and vision benefits and support for moving expenses available.

DEADLINE AND APPLICATION PROCESS: Applicants should submit a curriculum vitae, statement of teaching philosophy, description of research interest, an official copy of most recent transcripts, contact information for three references, and have three letters of recommendation sent separately. Review of applications will begin January 15, 2014, and will continue until the position is filled. The position will start in September 2014.

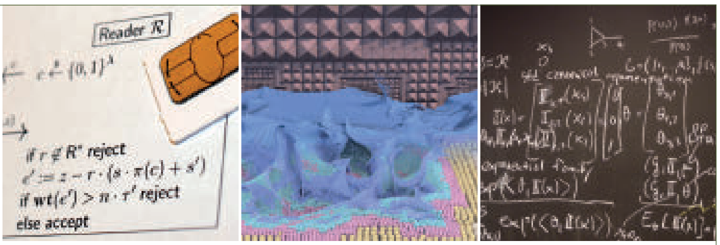
Please send all materials to:
Dr. Kerstin Voigt, Director
School of Computer Science and Engineering
California State University San Bernardino
5500 University Parkway
San Bernardino, CA 92407
Email Address: kvoigt@csusb.edu

Carnegie Mellon University
Computer Science Department
Teaching Track Positions

Applications are invited for two teaching-track positions in Computer Science, beginning Fall 2014. This is a renewable, career-oriented position with an initial appointment for three years. We seek highly qualified applicants with a strong commitment to excellence in teaching and the ability to teach at all levels in the undergraduate curriculum.

Applicants for the position must have a Ph.D. in Computer Science or a related field, and demonstrated excellence in teaching Computer Science courses. Teaching-track appointments are typically at the rank of Assistant Teaching Professor, with the possibility of promotion to the ranks of Associate Teaching Professor and Teaching Professor. None of these ranks are tenured; applicants seeking a tenure-track position at a research university are therefore not a good match for these positions.

In order to receive full consideration, applicants should submit a letter of application, curriculum vitae, a statement of teaching philosophy, and the names and email addresses of three or

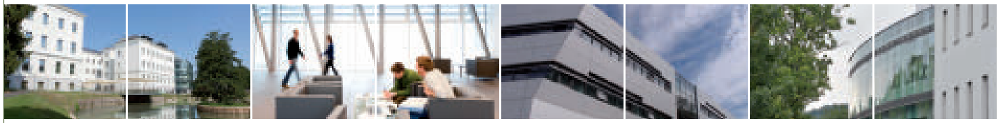


Are you

a graduate of Natural Sciences, Mathematics, or Computer Science?
Looking for a chance to start your scientific career as a PhD student?
Interested in crossing scientific boundaries?

Deadline for students wishing to enter the program in the fall of 2014 is **January 15, 2014**
For further information, please consult **www.ist.ac.at/gradschool**

IST Austria is committed to equality and diversity.



IST AUSTRIA
Institute of Science and Technology

ENGINEERING AT ILLINOIS

DRIVE YOUR VISION

ENDOWED CHAIRS AND PROFESSORSHIPS IN BIG DATA

Big Data will be the engine for the next generation of discoveries. As one of the Top 5 programs in the world, Engineering at Illinois has a head start and we plan to keep it. Thanks to the $100-million Grainger Engineering Breakthroughs Initiative, we're creating more than 35 new endowed professorships and chairs in Big Data and other fields. Applications and nominations are being accepted now. If you're ready to drive the future of Big Data, Illinois is the place for you.

GraingerInitiative.engineering.illinois.edu

ILLINOIS

Illinois is an Affirmative Action/Equal Opportunity Employer. www.inclusiveillinois.illinois.edu.
Full consideration will be given to applications and nominations received by December 16, 2013.

more individuals whom the applicant has asked to provide letters of reference. Applicants should arrange for reference letters to be sent directly to the contact below. This information should be sent by January 31, 2014, to the contact listed below.

Additionally, applicants are encouraged to submit a video sample of their teaching, demonstrating their excellence.

Please send your applications and accompanying materials to

Dr. Klaus Sutner
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
email: sutner@cs.cmu.edu

Carnegie Mellon is an affirmative action/equal opportunity employer and we invite and encourage applications from women and underrepresented minorities.

Chapman University

Assistant/Associate Professor of Computer Science or Software Engineering

Chapman University seeks applications for **two** tenure track Software Engineering/Computer Science faculty positions. The ideal candidate will be able to support graduate programs in **Computational and Data Science** as well as undergraduate programs in **Computer Science** and **Software Engineering**. For details see http://web.chapman.edu/jobs/job.aspx?id=579

Colorado State University

Department of Computer Science

Tenure-Track Faculty in Computer Science

Colorado State University is accepting applications for a tenure-track assistant professor in Computer Science, beginning fall 2014. Only candidates in bioinformatics/computational biology will be considered. This position is part of a university-wide effort to recruit additional faculty in biosciences and bioinformatics. Applications must be received by December 16, 2013. Submit materials at http://cns.natsci.colostate.edu/employment/Compsci/. CSU is an EO/EA/AA employer. Colorado State University conducts background checks on the final candidates.

Columbia University

Columbia Engineering's Department of Computer Science

Faculty Positions in Computer Science

Columbia Engineering's Department of Computer Science at Columbia University in New York City invites applications for tenure-track faculty positions. Appointments at the assistant professor (without tenure) and associate professor (without tenure) levels will be considered.

Applications are specifically sought in any of the areas that fall under the umbrella of Computer Science with particular emphasis on, but not limited to: Databases, Computational Biology, and Human-Computer Interaction.

Candidates must have a Ph.D. or its professional equivalent by the starting date of the appointment. Applicants for this position at the Assistant Professor and untenured Associate Professor level must have the potential to do pioneering research and to teach effectively.

The successful candidate should contribute to the advancement of the Department in these areas by developing an externally funded research program and contributing to the undergraduate and graduate educational mission of the Department, and is expected to establish multidisciplinary research and educational collaborations with academic departments and units across Columbia University. The Department is especially interested in qualified candidates who can contribute, through their research, teaching, and/or service, to the diversity and excellence of the academic community.

For additional information and to apply, please see: http://engineering.columbia.edu/faculty-job-opportunities. Applications should be submitted electronically and include the following: curriculum-vitae including a publication list, a research statement, a teaching statement, contact information for three people who can provide letters of recommendation, and up to three pre/reprints of scholarly work. The position will close no sooner than December 15, 2013, and will remain open until filled.

Applicants can consult www.cs.columbia.edu for more information about the Department.

Columbia is an affirmative action/equal opportunity employer with a strong commitment to the quality of faculty life.

FIU FLORIDA INTERNATIONAL UNIVERSITY

Florida International University is a comprehensive university offering 340 majors in 188 degree programs in 23 colleges and schools, with innovative bachelor's, master's and doctoral programs across all disciplines including medicine, public health, law, journalism, hospitality, and architecture. FIU is Carnegie-designated as both a research university with high research activity and a community-engaged university. Located in the heart of the dynamic south Florida urban region, our multiple campuses serve over 50,000 students, placing FIU among the ten largest universities in the nation. Our annual research expenditures in excess of $100 million and our deep commitment to engagement have made FIU the go-to solutions center for issues ranging from local to global. FIU leads the nation in granting bachelor's degrees, including in the STEM fields, to minority students and is first in awarding STEM master's degrees to Hispanics. Our students, faculty, and staff reflect Miami's diverse population, earning FIU the designation of Hispanic-Serving Institution. At FIU, we are proud to be 'Worlds Ahead'! For more information about FIU, visit ***fiu.edu***.

The School of Computing and Information Sciences at Florida International University seeks candidates for tenure-track and tenured faculty positions at all levels.

Open-Rank Tenure Track/Tenured Positions (Job ID# 506754)

We seek outstanding candidates in all areas of Computer Science and researchers in the areas of compilers and programming languages, computer architecture, databases, information retrieval and big data, natural language processing, and health informatics, are particularly encouraged to apply. Candidates from minority groups are encouraged to apply. Preference will be given to candidates who will enhance or complement our existing research strengths.

Ideal candidates for junior positions should have a record of exceptional research in their early careers. Candidates for senior positions must have an active and proven record of excellence in funded research, publications, and professional service, as well as a demonstrated ability to develop and lead collaborative research projects. In addition to developing or expanding a high-quality research program, all successful applicants must be committed to excellence in teaching at both the graduate and undergraduate levels. An earned Ph.D. in Computer Science or related disciplines is required.

Florida International University (FIU) is the state university of Florida in Miami. It is ranked by the Carnegie Foundation as a comprehensive, doctoral research university with high research activity. The School of Computing and Information Sciences (SCIS) is a rapidly growing program of excellence at the University, with 36 faculty members and over 1,800 students, including 80 Ph.D. students. SCIS offers B.S., M.S., and Ph.D. degrees in Computer Science, an M.S. degree in Telecommunications and Networking, and B.S., B.A., and M.S. degrees in Information Technology. SCIS has received approximately $19.6M in the last four years in external research funding, has 14 research centers/clusters with first-class computing infrastructure and support, and enjoys broad and dynamic industry and international partnerships.

HOW TO APPLY: Applications, including a letter of interest, contact information, curriculum vitae, academic transcript, and the names of at least three references, should be submitted directly to the **FIU Careers Website at *careers.fiu.edu***; refer to Job ID# 506754. The application review process will begin on January 1st, 2014, and will continue until the position is filled. Further information can be obtained from the School website ***http://www.cis.fiu.edu***, or by e-mail to ***recruit@cis.fiu.edu***.

FIU is a member of the State University System of Florida and is an Equal Opportunity, Equal Access Affirmative Action Employer.

Dartmouth College
Department of Computer Science
Assistant Professor of Computer Science: Computer Graphics/Digital Arts

The **Dartmouth College Department of Computer Science** invites applications for a tenure-track faculty position at the level of assistant professor. We seek candidates who will be excellent researchers and teachers in the areas of computer graphics and/or digital arts, although outstanding candidates in any area will be considered. We particularly seek candidates who will be integral members of the Digital Arts program and help lead, initiate, and participate in collaborative research projects both within Computer Science and involving other Dartmouth researchers, including those in other Arts & Sciences departments, Dartmouth's Geisel School of Medicine, and Thayer School of Engineering.

The department is home to 17 tenured and tenure-track faculty members and two research faculty members. Research areas of the department encompass the areas of systems, security, vision, digital arts, algorithms, theory, robotics, and computational biology. The Computer Science department is in the School of Arts & Sciences, and it has strong Ph.D. and M.S. programs and outstanding undergraduate majors. Digital Arts at Dartmouth is an interdisciplinary program housed in the Computer Science department, working with several other departments, including Studio Art, Theater, and Film and Media Studies. The department is affiliated with Dartmouth's M.D.-Ph.D. program and has strong collaborations with Dartmouth's other schools.

Dartmouth College, a member of the Ivy League, is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. Recreational opportunities abound in all four seasons.

With an even distribution of male and female students and over one third of the undergraduate student population members of minority groups, Dartmouth is committed to diversity and encourages applications from women and minorities.

To create an atmosphere supportive of research, Dartmouth offers new faculty members grants for research-related expenses, a quarter of sabbatical leave for each three academic years in residence, and flexible scheduling of teaching responsibilities.

Applicants are invited to submit application materials via Interfolio at http://apply.interfolio.com/23489. Upload a CV, research statement, and teaching statement, and request at least four references to upload letters of recommendation, at least one of which should comment on teaching. Email facsearch14@cs.dartmouth.edu with any questions.

Application review will begin November 1, 2013, and continue until the position is filled.

Dartmouth College
Department of Computer Science
Assistant Professor of Computer Science: Machine Learning

The **Dartmouth College Department of Computer Science** invites applications for a tenure-track faculty position at the level of assistant professor. We seek candidates who will be excellent researchers and teachers in the area of machine learning, although outstanding candidates in any area will be considered. We particularly seek candidates who will help lead, initiate, and participate in collaborative research projects both within Computer Science and involving other Dartmouth researchers, including those in other Arts & Sciences departments, Dartmouth's Geisel School of Medicine, Thayer School of Engineering, and Tuck School of Business.

The department is home to 17 tenured and tenure-track faculty members and two research faculty members. Research areas of the department encompass the areas of systems, security, vision, digital arts, algorithms, theory, robotics, and computational biology. The Computer Science department is in the School of Arts & Sciences, and it has strong Ph.D. and M.S. programs and outstanding undergraduate majors. The department is affiliated with Dartmouth's M.D.-Ph.D. program and has strong collaborations with Dartmouth's other schools.

Dartmouth College, a member of the Ivy League, is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. Recreational opportunities abound in all four seasons.

With an even distribution of male and female students and over one third of the undergraduate student population members of minority groups, Dartmouth is committed to diversity and encourages applications from women and minorities.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Assistant Professorships (Tenure Track) in Computer Science

The Department of Computer Science (www.inf.ethz.ch) at ETH Zurich invites applications for assistant professorships (Tenure Track) in the areas of:

– Computer Systems
– Software Engineering
– Information Systems (with emphasis on Big Data)

For candidates with exceptional research accomplishments also applications for a full professorship will be considered.

The department offers a stimulating and well-supported research and teaching environment. Collaboration in research and teaching is expected both within the department and with other groups of ETH Zurich and related institutions.

Applicants should have internationally recognized expertise in their field and pursue research at the forefront of Computer Science. Successful candidates should establish and lead a strong research program. They will be expected to supervise Ph.D. students and teach both undergraduate level courses (in German or English) and graduate level courses (in English).

Assistant professorships have been established to promote the careers of younger scientists. The initial appointment is for four years with the possibility of renewal for an additional two-year period and promotion to a permanent position.

Please apply online at www.facultyaffairs.ethz.ch

Applications should include a curriculum vitae, a list of publications, a statement of research and teaching interests and the names of at least three referees. The letter of application should be addressed **to the President of ETH Zurich, Prof. Dr. Ralph Eichler. The closing date for applications is 15 January 2014.** ETH Zurich is an equal opportunity and family friendly employer and is further responsive to the needs of dual career couples. In order to increase the number of women in leading academic positions, we specifically encourage women to apply.

To create an atmosphere supportive of research, Dartmouth offers new faculty members grants for research-related expenses, a quarter of sabbatical leave for each three academic years in residence, and flexible scheduling of teaching responsibilities.

Applicants are invited to submit application materials via Interfolio at http://apply.interfolio.com/23502. Upload a CV, research statement, and teaching statement, and request at least four references to upload letters of recommendation, at least one of which should comment on teaching. Email facsearch14@cs.dartmouth.edu with any questions.

Application review will begin November 1, 2013, and continue until the position is filled.

Dartmouth College

Department of Computer Science

Assistant Professor of Computer Science: Theory/Algorithms

The **Dartmouth College Department of Computer Science** invites applications for a tenure-track faculty position at the level of assistant professor. We seek candidates who will be excellent researchers and teachers in the area of theoretical computer science, including algorithms, although outstanding candidates in any area will be considered. We particularly seek candidates who will help lead, initiate, and participate in collaborative research projects both within Computer Science and involving other Dartmouth researchers, including those in other Arts & Sciences departments, Dartmouth's Geisel School of Medicine, Thayer School of Engineering, and Tuck School of Business.

The department is home to 17 tenured and tenure-track faculty members and two research faculty members. Research areas of the department encompass the areas of systems, security, vision, digital arts, algorithms, theory, robotics, and computational biology. The Computer Science department is in the School of Arts & Sciences, and it has strong Ph.D. and M.S. programs and outstanding undergraduate majors. The department is affiliated with Dartmouth's M.D.-Ph.D. program and has strong collaborations with Dartmouth's other schools.

Dartmouth College, a member of the Ivy League, is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. Recreational opportunities abound in all four seasons.

With an even distribution of male and female students and over one third of the undergraduate student population members of minority groups, Dartmouth is committed to diversity and encourages applications from women and minorities.

To create an atmosphere supportive of research, Dartmouth offers new faculty members grants for research-related expenses, a quarter of sabbatical leave for each three academic years in residence, and flexible scheduling of teaching responsibilities.

Applicants are invited to submit application materials via Interfolio at http://apply.interfolio.com/23503. Upload a CV, research statement, and teaching statement, and request at least four references to upload letters of recommendation, at least one of which should comment on teaching. Email facsearch14@cs.dartmouth.edu with any questions.

Application review will begin November 1, 2013, and continue until the position is filled.

Georgia Institute of Technology

School of Computer Science

The **School of Computer Science** at the **Georgia Institute of Technology** invites applications for **tenure-track faculty positions** from all areas of computer science. We seek candidates that can add to or enhance the current research areas of the school, with particular interest for candidates in the systems area and candidates who bridge research areas. In order to receive full consideration, please submit your application online by December 9, 2013. Full details at http://scs.gatech.edu.

Georgia Tech is an Affirmative Action/Equal Opportunity Employer. Applications from women and under-represented minorities are strongly encouraged.

Georgia State University

Open Rank Tenure Track Positions in Transcultural Conflict and Violence

Georgia State University announces an open rank, tenure-track faculty position in Transcul-

FACULTY POSITIONS IN COMPUTER SCIENCE

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

The Computer Science Program in the Computer, Electrical, and Mathematical Sciences and Engineering Division at King Abdullah University of Science and Technology (KAUST) invites applications for faculty positions at all ranks (Assistant, Associate, and Full Professors).

KAUST is an international, graduate research university dedicated to advancing science and technology through interdisciplinary research, education, and innovation. Located on the shores of the Red Sea in Saudi Arabia, KAUST offers superb research facilities, and internationally competitive salaries. The university attracts top international faculty and students to conduct fundamental and goal-oriented research to address the world's pressing scientific and technological challenges.

Areas of interest are:

- Artificial Intelligence
- Data Management and Big Data: Storing, indexing and querying very large datasets using large parallel and distributed computing systems; mining and knowledge extraction from very large data
- Large Scale Data Mining and Knowledge Extraction
- Parallel and Distributed Systems
- Visual Computing (including but not limited to Visualization and Physics-based simulation)
- Bioinformatics and all related subfields (e.g. Computational Genomics, Synthetic Biology, Biological Networks)
- High Performance Computing
- Operating Systems

Further information can be obtained by visiting **http://apptrkr.com/398993**
Applicants should have a proven track record, relevant PhD degree, as well as the ability to establish a high impact research program and demonstrate commitment to teaching at the graduate level.

How to Apply:
You will be required to complete a brief application form and upload a single PDF file including:

- Complete curriculum vitae with a list of publications
- Research plan
- Statement of teaching interests
- Names and contact information for at least three references for an Assistant Professor position
- List with the names and affiliation of potential referees for Associate Professor and Full Professor positions

Applications received by **January 15, 2014** will receive full consideration and positions will remain open until filled.

tural Conflict and Violence to start August 2014 pending budgetary approval. Ph.D. required. The faculty member will join an interdisciplinary research team of faculty from Computer Science, Communication, Religious Studies, Political Science, English and the Middle East Institute who draw upon cultural studies, social science, and technical fields to address complex global security challenges. The home department for the position will depend on the candidate's scholarly area of expertise. A successful candidate from computer science, for example, might specialize in visual analytics, semantic image processing, semantic web data mining, natural language processing and/or multimedia database management, from political science might have experience in global experimental or survey methodologies in conflict regions, and from communication might bring skills in media analytics or interventions for resolving conflict.

Prospective candidates should bring national or international recognition to the initiative and their home department. The hired applicant must have an established track record of securing external funds, as well as demonstrated ability and interest for recruiting top quality Ph.D. candidates and interdisciplinary research collaborations.

Applicants should send a letter of interest, curriculum vita, and three letters of recommendation to Dr. Carol Winkler, Chair, Search Committee, College of Arts and Sciences, Georgia State University, P.O. Box 4038, Atlanta, GA 30302-4038. GSU is a unit of the University System of Georgia and an AA/EO employer. An offer of employment will be contingent on background verification. Review of applications will begin October 1, 2013 and remain open until the position is filled. For more information contact Carol Winkler (cwinkler@gsu.edu).

Hendrix College
Assistant Professor of Computer Science

Hendrix College invites applications to join our Computer Science program in August 2014 as a tenure-track Assistant Professor. The successful candidate will be committed to excellent teaching in a liberal arts environment and will sustain a research program involving undergraduates. For fullest consideration, submit an online application (https://academicjobsonline.org/ajo/jobs/3347) by December 1, 2013.

The Johns Hopkins University
Department of Computer Science
Tenure Track Positions in Computer Science

With the anticipated opening of our new building, Malone Hall, the **Department of Computer Science** at **The Johns Hopkins University** is planning to continue our substantial multi-year growth. We solicit applications for tenure track positions at all levels. The search is open to all areas of Computer Science. On behalf of the Johns Hopkins Information Security Institute (JHUISI), we particularly encourage applicants with an interest in Cryptography, Systems and Network Security, Cloud and Mobile Security, and Health-Related Applications Security to apply. Applicants in systems and networking are also strongly desired.

All applicants must have a Ph.D. in Computer Science or a related field and will be expected to establish a strong, independent, multidisciplinary, internationally recognized research program. Commitment to quality teaching at the undergraduate and graduate levels is required. The department is committed to building a diverse educational environment; women and minorities are especially encouraged to apply.

A more extensive description of our search and additional supporting information can be found at http://www.cs.jhu.edu/Search2014.More information on the department is available athttp://www.cs.jhu.edu.

Applicants should apply using the online application which can be accessed from http://www.cs.jhu.edu/apply. Applications should be received by December 15, 2013 for full consideration. Questions should be directed to fsearch@cs.jhu.edu. The Johns Hopkins University is an EEO/AA employer.

Faculty Search
Johns Hopkins University
Department of Computer Science
Room 224 New Engineering Building
Baltimore, MD 21218-2694
Fax: 410-516-6134
Phone: 410-516-8775
fsearch@cs.jhu.edu
http://www.cs.jhu.edu/apply

Penn Engineering

Tenured/Tenure-Track Faculty

The Department of Electrical and Systems Engineering of the School of Engineering and Applied Science at the **University of Pennsylvania** invites applications for tenured and tenure-track faculty positions at all levels. We are interested in candidates in all areas that enhance our research strengths in:

1. **Nanodevices and nanosystems** (nanophotonics, nanoelectronics, integrated devices and systems at nanoscale),
2. **Circuits and computer engineering** (analog and digital circuits, emerging circuit design, computer engineering, embedded systems), and
3. **Information and decision systems** (communications, control, signal processing, network science, markets and social systems).

Prospective candidates are strongly encouraged to address large scale societal problems in energy, transportation, health, economic and financial networks, critical infrastructure, and national security. Diversity candidates are strongly encouraged to apply. Interested persons should submit an online application at ***http://facultysearches.provost.upenn.edu/postings/40***. Review of applications will begin on December 1, 2013.

The University of Pennsylvania is an Equal Opportunity Employer. Minorities/Women/Individuals with Disabilities/Veterans are encouraged to apply.

JOIN THE INNOVATION.

معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute
Member of Qatar Foundation عضو في مؤسسة قطر

Qatar Computing Research Institute seeks talented scientists and software engineers to join our team and conduct world-class applied research focused on tackling large-scale computing challenges.

We offer unique opportunities for a strong career spanning academic and applied research in the areas of Arabic language technologies including natural language processing, information retrieval and machine translation, distributed systems, data analytics, cyber security, social computing and computational science and engineering.

Scientist applicants must hold (or will hold at the time of hiring) a PhD degree, and should have a compelling track record of accomplishments and publications, strong academic excellence, effective communication and collaboration skills.

Software engineer applicants must hold a degree in computer science, computer engineering or related field; MSc or PhD degree is a plus.

We also welcome applications for post doctoral researcher positions.

As a **national research institute** and proud member of Qatar Foundation, our research program offers a collaborative, multidisciplinary team environment endowed with a comprehensive support infrastructure.

Successful candidates will be offered a highly competitive compensation package including an attractive tax-free salary and additional benefits such as furnished accommodation, excellent medical insurance, generous annual paid leave, and more.

For full details about our vacancies and how to apply online please visit
http://www.qcri.qa/join-us/
For queries, please email
QFJobs@qf.org.qa

/QCRI.QA @QatarComputing QatarComputing QatarComputing www.qcri.qa

The Johns Hopkins University

The Information Security Institute

Information Security Institute, Tenure-Track Positions

The Information Security Institute at **The Johns Hopkins University (JHUISI)** is planning for substantial multi-year growth in the area of **Information Security and Cryptography**. Our administration is committed at the highest level to substantially growing our security program with multiple, concurrent tenure-track offers expected this year, at all levels. We particularly encourage candidates with research interests in Cryptography, Systems and Network Security, Cloud and Mobile Security, and Health-Related Applications Security. As one of the first university centers in the country, we are poised now for significant growth and are looking for several new faculty members to help us achieve that.

All applicants must have a Ph.D. in Computer Science or a related field and will be expected to establish a strong, independent, multidisciplinary, internationally recognized research program. Commitment to quality teaching at the undergraduate and graduate levels is required. The department is committed to building a diverse educational environment; women and minorities are especially encouraged to apply.

A more extensive description of our search and additional supporting information can be found at http://www.cs.jhu.edu/Search2014. More information on the department is available at http://www.cs.jhu.edu.

Applicants should apply using the online application which can be accessed from http://www.cs.jhu.edu/apply. Applications should be received by December 15, 2013 for full consideration. Questions should be directed to fsearch@cs.jhu.edu. The Johns Hopkins University is an EEO/AA employer.

Faculty Search
Johns Hopkins University
Department of Computer Science
Room 224 New Engineering Building
Baltimore, MD 21218-2694
Fax: 410-516-6134
Phone: 410-516-8775
fsearch@cs.jhu.edu
http://www.cs.jhu.edu/apply

Maharishi University of Management

Assistant/Associate Professor

Maharishi University of Management has a dynamic and growing M.S. Computer Science degree program with a specialty in software development and professional applications. The Computer Science Department at MUM invites applications for several full-time faculty positions in this innovative and unique program. Qualifications include Ph.D. in Computer Science (or closely related area), or M.S. and seven years of professional software development experience. Candidates will be considered for Assistant, Associate, or full Professor depending on experience and qualifications.

The primary responsibility is teaching computer science courses at the M.S. level. Applications will be reviewed as they are received until the position is filled. To apply, email curriculum vitae (pdf file) to cssearch2011@mum.edu. For further information, see http://www.mum.edu/ and http://mscs.mum.edu/. MUM is located in Fairfield, Iowa, and is an equal opportunity employer.

Max Planck Institute for Software Systems

Tenure-track / Tenured Positions

Applications are invited for tenure-track and tenured faculty positions in all areas related to the study, design, and engineering of software systems. These areas include, but are not limited to, security and privacy, embedded and mobile systems, social computing, large-scale data management and machine learning, programming languages and systems, software verification and analysis, parallel and distributed systems, storage systems, and networking.

A doctoral degree in computer science or related areas and an outstanding research record are required. Successful candidates are expected to build a team and pursue a highly visible research agenda, both independently and in collaboration with other groups. Senior candidates must have demonstrated leadership abilities and recognized international stature.

MPI-SWS, founded in 2005, is part of a network of eighty Max Planck Institutes, Germany's premier basic research facilities. MPIs have an established record of world-class, foundational research in the fields of medicine, biology, chemistry, physics, technology and humanities. Since

THE CHINESE UNIVERSITY OF HONG KONG

Applications are invited for:-

Faculty of Engineering
Professors / Associate Professors / Assistant Professors

(Ref. 1314/032(255)/2)

The Faculty of Engineering invites applications for several faculty posts at Professor / Associate Professor / Assistant Professor levels with prospect for substantiation in the interdisciplinary area of 'Big Data Analytics', which is a new strategic research initiative supported by the University's Focused Innovations Scheme and will complement current/planned strengths in different Departments under the Faculty. To lead the big data research initiative, senior academics in this area are particularly welcome to apply.

Currently, the Faculty is seeking candidates in the following areas:
- Theoretical, mathematical and algorithmic aspects in large data analytics;
- Large scale software systems and architecture in large data analytics;
- Application areas in large data analytics (including information analytics, network/Web analytics, financial analytics, or bio/medical analytics, etc.).

Applicants should have (i) a PhD degree; and (ii) a strong scholarly record demonstrating potential for teaching and research excellence. The appointees will be expected to (a) teach at both undergraduate and postgraduate levels; (b) develop a significant independent research programme with external funding; and (c) supervise postgraduate students. Appointments will normally be made on contract basis for three years initially, which, subject to performance and mutual agreement, may lead to longer-term appointment or substantiation later. Applications will be accepted until the posts are filled. Further information about the Faculty is available at *http://www.erg.cuhk.edu.hk*.

Salary and Fringe Benefits

Salary will be highly competitive, commensurate with qualifications and experience. The University offers a comprehensive fringe benefit package, including medical care, plus a contract-end gratuity for appointments of two years or longer, and housing benefits for eligible appointees. Further information about the University and the general terms of service for appointments is available at *http://www.per.cuhk.edu.hk*. The terms mentioned herein are for reference only and are subject to revision by the University.

Application Procedure

Please send full resume, copies of academic credentials, a publication list with abstracts of selected published papers, details of courses taught and evaluation results (if any), a research plan, a teaching statement, together with names of three to five referees, to the Dean, Faculty of Engineering by e-mail to *recruit-bda@erg.cuhk.edu.hk*. For enquiries, please contact Professor John C.S. Lui, the leader of this strategic initiative (e-mail: *cslui@cse.cuhk.edu.hk*). Applicants are requested to clearly indicate that they are applying for the posts under 'Big Data Analytics Initiative'. The Personal Information Collection Statement will be provided upon request. Please quote the reference number and mark 'Application – Confidential' on cover.

Department of Computer and Information Sciences

TEMPLE UNIVERSITY

Tenure Track Faculty

Applications are invited for a tenure-track junior faculty position in the Department of Computer and Information Sciences (CIS) at Temple University. Outstanding candidates in all areas of computer science will be considered, with priority given to candidates with solid systems and analytical skills and a research focus complementary to the existing CIS strengths in network/computer systems and data analytics. The successful applicant will hold a Ph.D. in Computer Science, Computer Engineering, or a closely related field. The successful applicant will also be expected to contribute to interdisciplinary research programs.

CIS has two undergraduate degree programs in Computer Science (CS) and in Information Science and Technology (IS&T), master's programs in CIS and IS&T, and a PhD program in CIS. CIS has undergone considerable growth in research in the past few years, during which research funding, publication rates, and the number of Ph.D. students has significantly increased. CIS is scheduled to move to a new building next year.

Please submit applications with all requested information online at http://academicjobsonline.org. For further information, including other openings, check http://www.cis.temple.edu or send email to search committee chair Dr. Eugene Kwatny at gkwatny@temple.edu. Review of candidates will begin in December and will continue until the position is filled.

1948, MPI researchers have won 17 Nobel prizes. MPI-SWS aspires to meet the highest standards of excellence and international recognition with its research in software systems.

To this end, the institute offers a unique environment that combines the best aspects of a university department and a research laboratory:
a) Faculty receive generous base funding to build and lead a team of graduate students and post-docs. They have full academic freedom and publish their research results freely.
b) Faculty supervise doctoral theses, and have the opportunity to teach graduate and undergraduate courses.
c) Faculty are provided with outstanding technical and administrative support facilities as well as internationally competitive compensation packages.

MPI-SWS currently has 10 tenured and tenure-track faculty and 40 doctoral and post-doctoral researchers. The institute is funded to support 17 faculty and up to 100 doctoral and post-doctoral positions. Additional growth through outside funding is possible. We maintain an open, international and diverse work environment and seek applications from outstanding researchers regardless of national origin or citizenship. The working language is English; knowledge of the German language is not required for a successful career at the institute.

The institute is located in Kaiserslautern and Saarbruecken, in the tri-border area of Germany, France and Luxembourg. The area offers a high standard of living, beautiful surroundings and easy access to major metropolitan areas in the center of Europe, as well as a stimulating, competitive and collaborative work environment. In immediate proximity are the MPI for Informatics, Saarland University, the Technical University of Kaiserslautern, the German Center for Artificial Intelligence (DFKI), and the Fraunhofer Institutes for Experimental Software Engineering and for Industrial Mathematics.

Qualified candidates should apply online at https://apply.mpi-sws.org. The review of applications will begin on December 15, 2013, and applicants are strongly encouraged to apply by that date; however, applications will continue to be accepted through January 2014.

The institute is committed to increasing the representation of minorities, women and individuals with physical disabilities in Computer Science. We particularly encourage such individuals to apply. You can find more information about the positions at: http://www.mpi-sws.org/index.php?n=careers/tenure-track

Mount Holyoke College
Computer Science Department
Tenure Track Assistant Professor

The Mount Holyoke College Computer Science Department invites applications for a tenure-track Assistant Professor position, beginning Fall 2014. Candidates must have (or expect) a Ph.D. in computer science or a related field by September 2014. To see full ad: https://www.mtholyoke.edu/sites/default/files/deanoffaculty/docs/CSTT2014-2015.pdf

North Carolina State University
Department of Computer Science
Faculty Position
Assistant/Associate/Full Professor

The Department of Computer Science at North Carolina State University (NCSU) seeks to fill a tenure-track faculty position in the areas of Security and Systems starting August 16, 2014.

Successful security or systems candidate must have a strong commitment to academic and research excellence, and an outstanding research record commensurate with the expectations of a major research university. Required credentials include a doctorate in Computer Science or a related field. While the department expects to hire at the Assistant Professor level, candidates with exceptional research records are encouraged to apply for a senior position. The department is one of the largest and oldest in the country. It is part of a top US College of Engineering, and has excellent and extensive ties with industry and government laboratories.). We have new institutes and centers in the domain of security, education informatics, and games. Department's research expenditures and recognition have been growing steadily as has the recognition of our impact in the areas of security, systems, software engineering, educational informatics, networking, and games. For example, we have one of the largest concentrations of NSF Early Career Award winners (24 of our current or former faculty have received one).

NCSU is located in Raleigh, the capital of North Carolina, which forms one vertex of the

ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to *acmmediasales@acm.org*. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will by typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Rates: $325.00 for six lines of text, 40 characters per line. $32.50 for each additional line after the first six. The MINIMUM is six lines.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact: *acmmediasales@acm.org*

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: *http://jobs.acm.org*

Ads are listed for a period of 30 days.

For More Information Contact:

ACM Media Sales
at 212-626-0686 or
acmmediasales@acm.org

Director, Health Care Engineering Systems Center & Professor
College of Engineering
University of Illinois at Urbana-Champaign

National/International Search

The College of Engineering at the University of Illinois at Urbana-Champaign seeks interested applicants for a tenured faculty position as Director of the Health Care Engineering Systems Center at the Urbana campus. The Health Care Engineering Systems Center is an exciting new initiative being launched in partnership with the Jump Simulation and Education Center in Peoria, Illinois. We seek a Director with the vision and leadership skills to help create the premier center for providing clinical immersion and collaborations for engineering students with clinicians; develop new technologies and cyber-physical systems; enhance medical training and practice; and in collaboration with key partners, drive the training of medical practitioners of the future. For more information regarding the Jump Simulation and Education Center, please visit the following website: http://jumpsimulation.org/

Candidates are expected to hold a tenured faculty position at the level of full Professor (or an endowed position) in Industrial and Enterprise Systems Engineering (ISE), Bioengineering, Computer Science, Mechanical Engineering or any related Department. They must have a proven record of accomplishment in research, teaching, and leadership in the areas of interest related to this initiative. They must also show evidence of strong collaborative, administrative, and communication skills in order to effectively interact with college and campus administration, colleagues at the Jump Simulation and Education Center, clinicians, faculty, staff, and students.

To ensure full consideration, applications should be received by **January 15, 2014**, but applications will be accepted until the position is filled. Salary is commensurate with skills and experience. To apply for this position, please create your candidate profile at **http://jobs.illinois.edu.** For further information regarding application procedures, please contact Linda Morris at lsmorris@illinois.edu or 217-333-2511.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Illinois is an AA-EOE

world-famous Research Triangle Park (RTP). RTP is an innovative environment, both as a metropolitan area with one of the most diverse industrial bases in the world, and as a center of excellence promoting technology and science. The Research Triangle area is routinely recognized in nationwide surveys as one of the best places to live in the U.S. We enjoy outstanding public schools, affordable housing, and great weather, all in the proximity to the mountains and the seashore.

Applications will be reviewed as they are received. The positions will remain open until suitable candidates are identified. Applicants should submit the following materials online at http://jobs.ncsu.edu (reference position number 00001074) cover letter, curriculum vitae, research statement, teaching statement, and names and complete contact information of four references, including email addresses and phone numbers. Candidates can obtain information about the department and its research programs, as well as more detail about the position advertised here at http://www.csc.ncsu.edu/. Inquiries may be sent via email to: security-systems-search@csc.ncsu.edu.

NCSU is an equal opportunity and affirmative action employer. In addition, NCSU welcomes all persons without regard to sexual orientation or genetic information. Individuals with disabilities desiring accommodations in the application process should contact (919) 515-3148.

Oregon State University
School of Electrical Engineering and Computer Science
Assistant/Associate/Full Professor tenure-track Position in Computer Science

The School of Electrical Engineering and Computer Science at Oregon State University invites applications for up to six tenure-track positions in Computer Science. We seek candidates with a commitment to quality teaching and with demonstrated research strengths in the areas of databases, software engineering, computer security, distributed systems, and computer vision. Applicants should demonstrate a commitment to collaboration with other research groups in the School of EECS, with other departments at Oregon State, and with other universities.

The School of EECS supports a culture of energetic collaboration, and the faculty are committed to excellence in both education and research. With 48 tenure/tenure-track faculty, we enroll 180 Ph.D., 175 MS and 2000 undergraduate students. OSU is recognized for its “very high research activity” by the Carnegie Foundation for the Advancement of Teaching. The School of EECS is housed in the Kelley Engineering Center, a green building designed to support collaboration among faculty and students across campus. Corvallis is a small college town renowned for its high quality of life. http://oregonstate.edu/jobs/

The College at Brockport, State University of New York
Computer Science Department
Assistant Professor

Applications are invited for a tenure-track **Assistant Professor** position in Information Systems in the Computer Science Department beginning Fall 2014. Doctoral degree in Information Systems, or doctoral degree in a closely related discipline with master’s level expertise in Information Systems is required. ABD candidates considered.

Preference will be given to candidates with expertise in Information Assurance, Mobile and Cloud Computing, Human-Computer Interaction and Multimedia, Data Management.

Apply online at www.brockportrecruit.org by January 12, 2014. For more information visit www.brockport.edu/cs/vacancy.html. AA/EOE

Toyota Technological Institute at Chicago
Faculty Positions at All Levels

Toyota Technological Institute at Chicago is a philanthropically endowed degree-granting institute for computer science located on the University of Chicago campus. Applications are being accepted in all areas, but we are particularly interested in machine learning, speech processing, computational linguistics, computer vision, computational biology and optimization. Positions and visiting positions are available at all ranks, and we have a large number of three year limited term positions currently available. For all positions we require a Ph.D. Degree or Ph.D. candidacy, with the degree conferred prior to date of hire.

Toyota Technological Institute at Chicago is an Equal Opportunity Employer

Trinity University
Assistant Professor

Trinity University seeks applications for a tenure-track assistant professor position in the Department of Computer Science, to commence August 2014.

Candidates must have a Ph.D. in Computer Science or a related area. Our new colleague will teach introductory and advanced courses in CS, advise first-year students and majors, and be willing to include undergraduates in research activities. He/she will participate in ongoing development of the CS curriculum and will be active in research and professional activities. The teaching load is 9 student contact hours (3 courses) per semester.

Trinity is an independent, coeducational, primarily undergraduate and residential university founded in 1869. Enrollment is approximately 2400 from all areas of the US and many foreign countries. Trinity has highly selective admission standards and occupies an attractive campus overlooking downtown San Antonio, a city rich in heritage and ethnic diversity.

Perhaps unexpectedly, San Antonio has a particularly lively computing community! Indeed, San Antonio is second in the nation (only to the Washington, D.C. area) in cyber-security. While the position is open to all areas, a new colleague would find San Antonio to be a particularly good fit for information assurance & security, bioinformatics, or cloud computing.

The Department, 44 years old, is well integrated into campus life. Our students are among the best in the University and secure excellent offers for jobs and graduate schools. Early in 2014 the Department is scheduled to move into a new Center for the Sciences and Innovation, housing all the STEM departments and Entrepreneurship. Additional information may be found at

http://web.trinity.edu/x9627.xml
http://web.trinity.edu/x13575.xml

Please send a letter of application; CV; teaching and research interests; and names and contact information of at least three references to

Paul Myers, Chair
Department of Computer Science
Trinity University
1 Trinity Place
San Antonio, Texas 78212-7200
(210) 999-7398
pmyers@trinity.edu

Early review of applications will begin December 9, 2013.

Trinity University is an Equal Opportunity Employer.

Tufts University
Dept. of Computer Science
Faculty Position in Systems or Computational/Systems Biology

The Department of Computer Science at Tufts University invites applications for a faculty appointment to begin in September 2014. We welcome applicants of all ranks with research that strengthen our departmental, within-school, and cross-school research programs in the areas of (a) Systems, and (b) Computational/Systems Biology.

For more information, please visit <http://www.cs.tufts.edu/>. Inquiries should be emailed to cssearch@cs.tufts.edu. Review of applications begins December 9, 2013.

Tufts University is an Affirmative Action/Equal Opportunity employer. We are committed to increasing the diversity of our faculty. Members of underrepresented groups are strongly encouraged to apply.

University College London (UCL)
Department of Computer Science
Faculty Position

The Department of Computer Science at University College London (UCL) invites applications for a faculty position in the areas of Computer Systems and Networking. We seek world-class talent; candidates must have an outstanding research track record.

Areas of interest for this position include operating systems, systems security, distributed systems, networking, and their intersection, with an emphasis on experimental system-building. Appointments will be made at the rank of Lecturer, Senior Lecturer, or Reader (equivalent to Assistant Professor, junior Associate Professor, and senior Associate Professor, respectively, in the US system), commensurate with qualifications.

Candidates must hold an earned Ph.D. in Computer Science or a closely related field by the time they begin their appointment. They will be evaluated chiefly on the significance and novelty of their research to date, and their promise for leading a group in a fruitful program of research. They must also demonstrate a zest for innovative and challenging teaching at the graduate and undergraduate levels. A proven record of ability to manage time and evidence of ability to teach and to supervise academic work by undergraduates, masters, and doctoral students are desirable. Our department is a highly collaborative environment, and we seek future colleagues who enjoy working collaboratively. Candidates should further be committed to public communication, and to UCL's policy of equal opportunity, including working harmoniously with colleagues and students of all cultures and backgrounds.

Since 1973, when UCL CS became the first ARPAnet node outside the United States, the department has been a leading centre for networking research, as demonstrated by its long-standing strong presence in the SIGCOMM conference proceedings.

UCL, the third oldest university in England, and the first to admit students without regard for their race or creed, sits a few blocks from the British Museum in central London, a vibrant metropolis of 8.3 million that offers an ever-renewing panoply of cultural and culinary choices, and is a budget airfare away from all of continental Europe.

Further details about UCL CS, the posts, and how to apply may be found at http://www.cs.ucl.ac.uk/vacancies

All application materials must reach UCL by the 17th of January, 2014.

We particularly welcome female applicants and those from an ethnic minority, as they are under-represented within UCL at these levels.

UCL – Taking Action for Equality.

The University of Alabama in Huntsville
Department of Computer Science
Assistant Professor

The **Department of Computer Science** of **The University of Alabama in Huntsville (UAH)** invites applicants for a tenure-track faculty position at the **Assistant Professor** level (or possibly at Associate Professor for experienced, exceptional candidates). We require someone in entertainment computing, in particular in game development/animation, able to help us build out our portfolio in this area.

The position is one part of a multi-year strategic investment in entertainment computing and arts by the university. A Ph.D. in computer science or a closely related area is required. The successful candidate will have a strong interest in aiding our coordination with peers in disciplines such as animation arts, music, etc., and have a strong academic background, be able to carry out research in areas typical for graphics and serious games academic conferences, and be keen on undergraduate education. A commitment to keeping abreast of games technology is an ongoing expectation for the hire. Professional experience in graphics/animation, especially related to game development, is a plus.

The department has a strong commitment to excellence in teaching, research, and service; the hire should have good communication, strong teaching potential, and research accomplishments.

UAH is located in an expanding, high technology area, next door to one of the largest research parks in the nation. Nearby are the NASA Marshall Space Flight Center, the Army's Redstone Arsenal, and many high-tech industries. UAH also has an array of research centers, including in information technology, modeling and simulation, etc. In short, collaborative research opportunities are abundant, and many well-educated and highly technically skilled people are in the area. There is also access to excellent public schools and inexpensive housing.

UAH has approximately 7500 students. UAH Computer Science offers the BS, MS, and PhD degrees in Computer Science and the MS and PhD degrees in modeling and simulation. Approximately 200 undergraduate majors and 175 graduate students are associated with the unit. Faculty research interests are many—areas closely related to this position include visualization, graphics, multimedia, AI, image processing, pattern recognition, mobile computing, and distributed systems. Recent NSF figures indicate the department ranks 30th in the nation in overall federal research funding.

Interested parties should submit a detailed resume with references to Chair, Search Committee, Dept. of Computer Science, The University of Alabama in Huntsville, Huntsville, AL 35899. Qualified female and minority candidates are encouraged to apply. Initial review of applicants will begin in January 2014 and will continue until a suitable candidate is found. UAH is an equal opportunity/affirmative action institution.

University of Central Florida
Computer Science
Tenure-Track Faculty Positions

The Computer Science Division in the College of Engineering and Computer Science at UCF is looking for exceptional tenure-track faculty at all levels, especially at senior levels. We are interested in hiring candidates with an established record of high-impact research and publication in: *big data, human-computer interaction, computer security, mobile computing, software engineering,* and *theory of computing,* though we will also consider remarkable candidates in other areas. We offer a competitive salary and start up package with generous benefits.

All applicants must have a Ph.D. from an accredited institution in an area appropriate to Computer Science and a strong commitment to the academic process, including teaching, scholarly publications, and sponsored research. Successful candidates will have a record of high-quality publications and be recognized for their expertise and the impact of their research.

Computer Science at UCF has a rapidly-growing educational and research program with nearly $4.6 million in research contracts and expenditures annually and over 179 graduate students, including 113 Ph.D. students. Computer Science has strong areas of research in Computer Vision, Machine Learning, Virtual and Mixed Reality, and HCI. More information about the Computer Science Division can be found at http://www.cs.ucf.edu/.

Research sponsors include NSF, NIH, NASA, DOT, DARPA, ONR, and other agencies of the DOD. Industry sponsors include AMD, Boeing, Canon, Electronic Arts, General Dynamics, Harris, Hitachi, Intel, Lockheed Martin, Oracle, SAIC, Symantec, Toyota USA, and Walt Disney World, as well as local startups.

UCF has about 60,000 students and is the nation's second largest university. Located in Orlando, UCF is at the center of the I-4 High Tech Corridor. The corridor has an excellent industrial base that includes: software, defense, space, simulation and training, and a world-renowned entertainment industry. Adjacent to UCF is a thriving research park that hosts more than 100 high-technology companies and the Institute for Simulation and Training. The Central Florida area is designated by the State of Florida as the Center of Excellence in Modeling and Simulation. UCF also has an accredited medical school, which opened in 2009. Great weather, easy access to the seashore, one of the largest convention centers in the nation, and one of the world's best airports are just a few features that make Orlando an ideal location. UCF is an Equal Opportunity/Affirmative Action employer. Women and minorities are particularly encouraged to apply.

To submit an application, please go to: http://www.cs.ucf.edu/facsearch

University of Colorado, Boulder
Assistant Professor

The Department of Computer Science at the University of Colorado Boulder seeks outstanding candidates for a tenure-track position in Cyber-Physical Systems. The opening is targeted at the level of Assistant Professor, although outstanding senior candidates at higher ranks may be considered.

We seek candidates whose primary research areas include verification, machine learning, robotics or embedded systems, and whose research addresses challenges in application areas such as computational material science, avionics, space systems, renewable energy, manufacturing, robotics or bio-medical systems. Candidates should integrate physical systems with relevant computational techniques, show promise of excellence in both research and teaching, and want to create and lead a highly visible collaborative research program.

Applications must be submitted on-line at http://www.jobsatcu.com/postings/73109

The University of Colorado is an Equal Opportunity/Affirmative Action employer.

University of Illinois at Chicago
Computer Science Department
Multiple Faculty Positions

The Computer Science Department at the University of Illinois at Chicago invites applications for multiple tenure-track positions in Computer Systems, broadly defined, at the rank of Assistant Professor. Exceptional candidates at other ranks may also be considered. Candidates in related areas whose research evidences a strong empirical focus are also encouraged to apply.

The University of Illinois at Chicago (UIC) ranks among the nation's top 50 universities in federal research funding and is ranked the fourth best U.S. University under 50 years old by Times Higher Education. The Computer Science Department is a leading department within the University. The Computer Science Department has 24 tenure-track faculty members, and offers BS, MS and PhD degrees. Our faculty includes two ACM Fellows and nine NSF CAREER award recipients. We have annual research expenditures of $9.4M, primarily federally funded. UIC is an excellent place for interdisciplinary research: UIC houses the largest medical school in the country, and our faculty are engaged in many cross-departmental collaborations with faculty from engineering, health sciences, social sciences, urban planning, and the business school.

Chicago epitomizes the modern, livable, vibrant city. Located on the shore of Lake Michigan, it offers an outstanding array of recreational and cultural experiences. As the birthplace of the modern skyscraper, Chicago boasts one of the world's tallest and densest skylines, combined with an extensive system of parks and public transit. Yet the cost of living, whether in an 85th floor condominium downtown or on a tree-lined street in one of the nation's finest school districts, is surprisingly low.

Applications must be submitted at https://jobs.uic.edu/. Please include a resume, teaching and research statements, and names and addresses of at least three references in the online application. Applicants needing additional information may contact the Faculty Search Chair at search@cs.uic.edu. For fullest consideration please apply by Nov. 25, 2013. The University of Illinois at Chicago is an Affirmative Action/Equal Opportunity Employer.

University of Illinois at Urbana-Champaign

Department of Electrical and Computer Engineering (ECE)
Faculty Positions

The Department of Electrical and Computer Engineering (ECE) at the University of Illinois at Urbana-Champaign invites applications for **faculty positions** at all areas and levels in Computing, broadly defined, with particular emphasis on big data, including complex data analysis and decision making; scalable hardware and software systems; parallel, high-performance, and energy-efficient computing; reliable and secure computing; bioinformatics and systems biology; and networking and distributed computing, among other areas. From the transistor and the first computer implementation based on von Neumann's architecture to the Blue Waters petascale computer – the fastest computer on any university campus, ECE Illinois faculty have always been at t he forefront of computing research and innovation. Applications are encouraged from candidates whose research programs specialize in traditional, nontraditional, and interdisciplinary areas of electrical and computer engineering. The department is engaged in exciting new and expanding programs for research, education, and professional development, with strong ties to industry.

Qualified senior candidates may also be considered for tenured full Professor positions as part of the Grainger Engineering Breakthroughs Initiative (http://graingerinitiative.engineering.illinois.edu), which is backed by a $100-million gift from the Grainger Foundation to support research in big data and bioengineering, broadly defined. In addition, the University of Illinois is home to Blue Waters - one of the most powerful supercomputers in the world, supported by the National Science Foundation and developed and operated by the University of Illinois' National Center for Supercomputing Applications. Qualified candidates may be hired as Blue Waters Professors who will be provided substantial allocations on and expedited access to the supercomputer. To be considered as a Blue Waters Professor, candidates need to mention Blue Waters as one of their preferred research areas in their online application, and include a reference to Blue Waters in their cover letter.

Please visit http://jobs.illinois.edu to view the complete position announcement and application instructions. Full consideration will be given to applications received by December 15, 2013.

Illinois is an AA-EOE. www.inclusiveillinois.illinois.edu

University of Kentucky

Computer Science Department
Assistant Professor

The **University of Kentucky Computer Science Department** expects to hire an **Assistant Professor** to begin employment in August of 2014. Candidates must have earned a PhD in Computer Science or closely related field at the time employment begins. Applications are now being accepted.

Review of credentials will begin immediately and continue until the position is filled.

The department seeks to hire energetic researcher/educators who are interested in the application of advanced computing to challenging and relevant problems. Our faculty undertake interdisciplinary research, working with other departments including statistics, biology, linguistics, electrical engineering, computer engineering, and the humanities. We therefore favor researchers who can collaborate to solve problems involving multiple disciplines. All areas of computing will be considered, but a focus area related to software systems, big data, multimedia and/or imaging is preferred. These areas tie to our department's Laboratory for Advanced Networking, Center for Visualization and Virtual Environments, and Software Engineering. It is possible that the right candidate could be eligible for a named professorship at this level.

To apply, a University of Kentucky Academic Profile must be submitted at ukjobs.uky.edu/applicants/Central?quickFind=245523

For more detailed information about this position, go to www.cs.uky.edu/opportunities/faculty.

The University of Kentucky is an equal opportunity employer and especially encourages applications from minorities and women.

University of Maryland, Baltimore County (UMBC)

An Honors University in Maryland
Information Systems Department
Assistant Professor

The Information Systems Department at UMBC invites applications for two tenure-track faculty positions at the **Assistant Professor** level starting August 2014. We are searching for candidates with research interests in any of the following areas: Artificial Intelligence/Knowledge Management, Database/Data Mining, Human Centered Computing, Software Engineering, and Health Information Technology. Candidates must have earned a PhD in Information Systems or a related field no later than August 2014.

Candidates should be engaged in research that spans two or more of these areas, with one of those areas ideally being Health Information Technology. Preference will be given to those who can collaborate with current faculty. Candidates for both positions should have a strong potential for excellence in research, the ability to develop and sustain an externally funded research program, and the ability to contribute to our graduate and undergraduate teaching mission.

The Department offers undergraduate degrees in Information Systems and Business Technology Administration. Graduate degree programs, MS and PhD, are offered in both Information Systems and Human-Centered Computing, including an innovative online MS in IS program. Consistent with the UMBC vision, the Department has excellent teaching facilities, state-of-the-art laboratories, and outstanding technical support. UMBC's Technology Center, Research Park, and Center for

Entrepreneurship are major indicators of active research and outreach. Further details on our research, academic programs, and faculty can be found at http://www.is.umbc.edu/. Members of under-represented groups including women and minorities are especially encouraged to apply.

Applications will not be reviewed until the following materials are received: a cover letter, a one-page statement of teaching interests, a one to two-page statement of research interests, one or more sample research papers, and a CV. Applicants should also arrange to have three letters of recommendation sent to the department as soon as possible. Electronic submission of materials as PDF documents is preferred. Electronic copies should be sent to bmorris@umbc.edu. Copies can also be sent to: Dr. Aryya Gangopadhyay, Chair of Faculty Search Committee, Information Systems Department, UMBC, 1000 Hilltop Circle, Baltimore, MD 21250-5398. For inquiries, please contact Barbara Morris at (410) 455-3795 or bmorris@umbc.edu.

Review of applications will begin immediately and will continue until the positions are filled. These positions are subject to the availability of funds.

UMBC is an Affirmative Action/Equal Opportunity Employer and welcomes applications from minorities, women and individuals with disabilities.

University of Maryland, Baltimore County (UMBC)

Computer Science and Electrical Engineering Department
Two Tenure Track Assistant Professor Positions, Computer Science

We invite applications for two tenure track positions in Computer Science at the rank of **Assistant Professor** to begin in August 2014. All areas will be considered, but we are especially interested in candidates in systems, security, or data analytics. Unusually strong candidates at the Associate Professor level will be considered. Submit a cover letter, brief statement of teaching and research experience and interests, CV, and three letters of recommendation. See http://csee.umbc.edu/about/jobs/ for more information about this search and concurrent searches for a tenure track position in Electrical and Computer Engineering and a Professor of the Practice position in Computer Science. UMBC is an AA/EOE.

University of Maryland, Baltimore County (UMBC)

An Honors University in Maryland
Information Systems Department - Lecturer

The Information Systems Department invites applications for a non-tenured Lecturer position with a start date of Spring Semester 2014. This individual is expected to be actively involved with the department's undergraduate programs and students. To that end it is anticipated that the successful candidate will have experience in the areas of teaching, advising and service. The department is multi-disciplinary, placing a strong emphasis on the theory and application of information systems. We are particularly interested in candidates with teaching interests and expertise in areas related to Management and Management Information Systems. Outstanding candidates in other areas will also be considered. At a minimum, candidates must have an earned MS in a relevant area. Candidates with significant industrial or governmental background are encouraged to apply.

The department offers a wide variety of courses at the Bachelors, Masters, and Doctoral levels, has one of the highest IS enrollments in the USA, and offers the only Ph.D. in Information Systems in Maryland. For a list of our course offerings see: undergraduate courses. Further details on our programs and faculty may be found at http://www.is.umbc.edu/. Consistent with the UMBC vision, the department has excellent technical support and teaching facilities as well as outstanding laboratory space and state of the art technology.

Interested applicants should send a cover letter, a one-page statement of teaching interests and curriculum vitae, and three letters of recommendation to: Dr. Aryya Gangopadhyay, Chair of Lecturer Search Committee, Information Systems Department, UMBC, 1000 Hilltop Circle, Baltimore, MD 21250-5398. For inquiries, please call Barbara Morris at (410) 455-3795 or e-mail: bmorris@umbc.edu. Electronic submission of materials as PDF documents (sent to the preceding e-mail address) is preferred.

Review of applications will begin October 15, 2013 and will continue until the position is filled. This position is subject to the availability of funds.

UMBC is an Affirmative Action/Equal Opportunity Employer and welcomes applications from minorities, women and individuals with disabilities.

University of Michigan-Dearborn

Assistant/Associate Professor in Computer and Information Science

The University of Michigan-Dearborn (UM-Dearborn) is one of the three campuses of the University of Michigan. UM-Dearborn, a comprehensive university offering high quality undergraduate, graduate, professional and continuing education to residents of southeastern Michigan, attracts more than 9,000 students.

The campus is located on 200 acres of the original Henry Ford Estate. Dearborn is centrally located within one of America's largest business regions. The geographically diverse area provides faculty with a variety of urban, suburban, and rural areas within a reasonable commute, including Detroit, Detroit suburbs, and Ann Arbor.

Job Description: Assistant/Associate Professor in Computer and Information Science

The Department of Computer and Information Science (CIS) invites applications for a tenure-track, assistant/associate professor health informatics faculty position, starting September 1, 2014. Funding for this position has been approved.

The Department of CIS has 16 faculty members representing computer graphics and geometric modeling, database systems and data management, multimedia systems and gaming, networking, computer and network security, and software engineering. These areas of research are supported by several established labs and many of these areas are currently funded.

We offer several BS and MS degrees, and participate in several interdisciplinary degree programs, including an MS program in software engineering and a Ph.D. program in information systems engineering.

Qualifications

Qualified candidates should be doing research in areas of health informatics such as, but not restricted to, big data and analytics, data mining and machine learning, information and image retrieval, integration of heterogeneous health data sources, image analysis, spatio-temporal models, security and privacy of clinical data, streaming databases, or data visualization. Candidates will be expected to do scholarly and sponsored research, as well as teaching at both the undergraduate and graduate levels. Candidates at the associate professor rank should already have an established funded research program. Rank and salary will be commensurate with qualifications and experience. We offer competitive salaries and start-up packages.

Applicants should send a letter of intent, curriculum vitae, statements of teaching and research, evidence of any teaching performance, and at least three letters of recommendation to:

Dr. William Grosky (wgrosky@umich.edu)
Department of Computer and
Information Science
The University of Michigan-Dearborn
111 CIS Building
4901 Evergreen Road
Dearborn, Michigan 48128

In addition, applicants are requested to email materials available electronically to wgrosky@umich.edu. Review of applicants will begin January 1, 2014 and continue until the position has been filled.

The University of Michigan-Dearborn is an equal opportunity/affirmative action employer.

University of Minnesota-Twin Cities

Department of Computer Science and Engineering

Faculty Positions in Social Computing/ Social Media And Robotics

The Department of Computer Science and Engineering at the Univ. of Minnesota-Twin Cities invites applications from candidates for a faculty position in Social Computing/Social Media and multiple positions in Robotics. Specific topics of interest for the first position include social media analysis, crowdsourcing, geosocial systems, and data mining. Specific topics of interest for the Robotics positions include grasping, manipulation, legged locomotion, sensing and estimation, distributed decision making, machine learning, computer vision, robot design, algorithmic foundations, and embedded systems. We encourage applications from women and under- represented minorities. Candidates should have a PhD in Computer Science or a closely related discipline. The positions are open until filled, but for full consideration apply at www.cs.umn.edu/employment/faculty by December 13, 2013. The University of Minnesota is an equal opportunity employer and educator.

University of Nevada, Reno

CSE Department

Three Tenure-Track Assistant Professor Faculty Positions in CSE

The CSE Department invites applications for three **tenure-track Assistant Professor faculty positions in CSE**. One position is in the area of cybersecurity and one is in the area of advanced manufacturing with emphasis in robotics and automation, and digital manufacturing. Both positions are part of newly forming clusters, the first including departments in the School of Business and the College of Science, and the second encompassing several departments in the College of Engineering. The third position is open to all research areas, while candidates with expertise in embedded systems, computer architecture, high-performance computing for modeling, simulation and analysis, big data and cloud computing may be given preference. The department is dynamic and plans to strategically grow in the future. Our faculty have received NSF Career awards and are leaders in state-wide and multi-state multi-million dollar NSF awards. Our research is supported by NSF, DoD, NASA, Google, Microsoft and AT&T. The department's annual research expenditures have exceeded $2M in past years. We offer BS, MS, and Ph.D. degrees and have strong research and education programs in Intelligent Systems, Networks, Software Systems, and Games. Applicants must have a Ph.D. in Computer Science or Computer Engineering by July 1, 2014. Candidates must be strongly committed to excellence in research and teaching and should demonstrate potential for developing robust externally funded research programs. **Apply online at** https://www.unrsearch.com/postings/13561. Review of applications will begin January 15, 2014. Inquiries should be directed to Ms. Lisa Cody, lacody@unr.edu. UNR, Nevada's land-grant university, has close to 19,000 students. Reno, ranked #9 in the top 100 best places to live by Livability.com, is a half-hour drive to beautiful Lake Tahoe, an area that offers fantastic hiking and ski opportunities. San Francisco is within a four-hour's drive. UNR is an Equal Opportunity /Affirmative Action Institution.

University of Rochester

Department of Computer Science

Faculty Positions in Computer Science: Experimental Systems and Data Science

The **University of Rochester Department of Computer Science** seeks applicants for multiple tenure track positions in the broad areas of experimental systems and data science research (including but not exclusively focused on very large data-driven systems, machine learning and/or optimization, networks and distributed systems, operating systems, sustainable systems, security, and cloud computing). Candidates must have a PhD in computer science or a related discipline.

Apply online at
https://www.rochester.edu/fort/csc

Consideration of applications at any rank will begin immediately and continue until all interview slots are filled. Candidates should apply no later than January 1, 2014 for full consideration. Applications that arrive after this date incur a probability of being overlooked or arriving after the interview schedule is filled up.

The Department of Electrical and Computer Engineering (http://www.ece.rochester.edu/about/jobs.html) is also searching for a candidate broadly oriented toward data science. While the two searches are concurrent and plan to coordinate, candidates should apply to the department/s that best matches their academic background and interests.

The Department of Computer Science is a research-oriented department with a distinguished history of contributions in systems, theory, artificial intelligence, and HCI. We have a collaborative culture and strong ties to electrical and computer engineering, cognitive science, linguistics, and several departments in the medical center. Over the past decade, a third of the department's PhD graduates have won tenure-track faculty positions, and its alumni include leaders at major research laboratories such as Google, Microsoft, and IBM.

The University of Rochester is a private, Tier I research institution located in western New York State. It consistently ranks among the top 30 institutions, both public and private, in federal funding for research and development. The university has made substantial investments in computing infrastructure through the Center for Integrated Research Computing (CIRC) and the Health Sciences Center for Computational Innovation (HSCCI). Teaching loads are light and classes are small. Half of all undergraduates go on to post-graduate or professional education. The university includes the Eastman School of Music, a premiere music conservatory, and the University of Rochester Medical Center, a major medical school, research center, and hospital system. The greater Rochester area is home to over a million people, including 80,000 students who attend its 8 colleges and universities.

The University of Rochester has a strong commitment to diversity and actively encourages applications from candidates from groups underrepresented in higher education. The University is an Equal Opportunity Employer.

University of Tennessee at Martin
Assistant Professor of Computer Science

UTM seeks to fill a tenure-track position beginning 8/1/2014. A PhD in CS required. Candidates who are ABD may apply for a position as Lecturer, but a PhD is required for tenure and appointment as an Assistant Professor. Candidates must teach a variety of computer science courses typical of a 4-year computer science program. Applications may be made at http://www.utm.edu/departments/personnel//employment.php. Review of applications will begin 11/15/2013 and continue until the position is filled.

University of Texas at San Antonio
Department of Computer Science
Faculty Positions in Computer Science

The Department of Computer Science at The University of Texas at San Antonio invites applications for **multiple tenure/tenure-track positions at all levels**, starting Fall 2014. We are particularly interested in candidates in

▸ operating systems, distributed systems, or computer architecture at the assistant or associate professor level,
▸ big data/data science at the assistant or associate professor level, and
▸ cyber security (especially systems security) at the associate or full professor level.

Outstanding candidates in other areas will also be considered.

The Department of Computer Science currently has 22 faculty members and offers B.S., M.S., and Ph.D. degrees supporting a dynamic and growing program with 801 undergraduates and more than 190 graduate students, including 85 Ph.D. students. See http://www.cs.utsa.edu/fsearch for application instructions and additional information on the Department of Computer Science.

Screening of applications will begin on January 2, 2014 and will continue until the positions are filled or the search is closed. UTSA is an EO/AA Employer.

Chair of Faculty Search Committee
Department of Computer Science
The University of Texas at San Antonio
One UTSA Circle
San Antonio, TX 78249-0667
Phone: 210-458-4436

University of Wisconsin-Platteville
Department of Computer Science and Software Engineering
Assistant Professor of Computer Science and/or Software Engineering

The Department of Computer Science and Software Engineering at the **University of Wisconsin-Platteville** invites applications for two tenure-track faculty positions at the assistant professor level.

Candidates must have a Ph.D. in Software Engineering or Computer Science or a closely related field. Applicants must have a strong commitment to undergraduate education and teaching excellence.

Position starting August 19, 2014. See http://www.uwplatt.edu/pers/faculty.htm for complete announcement details and application instructions. AA/EEO employer.

University of Wisconsin-Stevens Point
Department of Computing and New Media Technologies
3 Tenure-line Faculty Positions

The fast growing and dynamic **Department of Computing and New Media Technologies** at the **University of Wisconsin-Stevens Point** invites applications for 3 tenure-line faculty positions at the rank of **assistant/associate professor** with appointments starting August 2014. The positions focus on: (1) **Software Development** - OO programming, algorithms and data structures, software engineering – analysis, design and testing of applications, networking, information assurance and cloud computing; (2) **Web Development** – server-side and client-side technologies, OO programming, rich Internet application technologies; and (3) **Multimedia** - multimedia design and development, digital video and audio production, aesthetics and human computer interaction (HCI), game development, 3D modeling. Check position descriptions and electronic application procedure at: http://www.uwsp.edu/equity/Pages/jobVacancies.aspx. Applicant screening will begin on January 2, 2014 and continue until the positions are filled.

U.S. Naval Academy
Computer Science Department
Assistant Professor

The U.S. Naval Academy's Computer Science Department invites applications for a tenure track position at the rank of **Assistant Professor**. This position is anticipated to begin in August 2014. A Ph.D. in Computer Science or closely related field is required.

The Computer Science Department offers majors in Computer Science and Information Technology, and contributes to a new interdisciplinary major in Cyber Operations. We currently have 100 CS majors, 70 IT majors and a faculty of 18. In addition to courses for CS, IT, and Cyber Operations majors, we also teach a course on cyber security to the entire freshman class.

Applicants must have a dedication to teaching, broad teaching interests, and a strong research program. Applications are encouraged from all areas of computing research.

The department is housed in a state of the art building overlooking the scenic Severn River. Our spaces provide outstanding office, laboratory, and research facilities for both students and faculty, including specialized labs for information assurance, networking, and robotics, as well as three micro-computing labs and two high performance computing labs.

The Naval Academy is an undergraduate institution located in historic downtown Annapolis, Maryland on the Chesapeake Bay. Roughly half of the faculty are tenured or tenure track civilian professors with Ph.D.s who balance teaching excellence with internationally recognized research programs. The remaining faculty are active duty military officers with Masters or Doctoral degrees. Each year the academy graduates roughly 1000 undergraduate students with majors in the sciences, engineering, and humanities. More information about the department and the Academy can be found at http://www.usna.edu/cs/ and http://www.usna.edu/.

Applicants should send a cover letter, teaching and research statements, curriculum vitae, and arrange for three letters of recommendation that address both teaching and research abilities to be sent to cssearch@usna.edu. Where possible, applicants are encouraged to include information that elaborates on the strength of their research program (such as conference acceptance rates or research awards).

Review of applications will begin immediately and will continue until the position is filled; apply by December 1, 2013 to ensure full consideration.

The United States Naval Academy is an Equal Opportunity Employer.

Washington State University
School of Electrical Engineering & Computer Science
Assistant/Associate/Full Professor

Washington State University invites applications for three full-time, tenure-track faculty positions in Computer Science with emphasis in machine learning. Please visit http://www.wsujobs.com/applicants/Central?quickFind=57998 for a complete description of the positions and qualifications. WSU is an EE/EO employer.

Worcester Polytechnic Institute
Computer Science Department
Tenure Track Faculty Members

Balance. Research & Teaching, Theory & Practice, Innovation & Fundamentals, Classes & Projects. Balance defines life at WPI. If you are that unique individual who strives for and achieves it, consider joining the faculty at WPI.

The Computer Science Department anticipates hiring one or more tenure track faculty for the Fall of 2014 whose expertise is in the following areas:

Computer Security including network, system, or software security, forensics and usable security; Data Sciences including databases, data mining, big data analytics and working with different application domains; Bioinformatics working in a cross-disciplinary program with faculty from Biology and Mathematics; Serious Games working with faculty in our Interactive Media & Game Development as well as our Learning Sciences & Technologies interdisciplinary programs; and Algorithms, particularly with connections to previously listed areas.

In addition to these specific areas, outstanding candidates in any area will receive full consideration. Candidates should have a PhD in Computer Science or a closely related field, and the potential for excellence in research and teaching.

Founded in 1865, WPI is one of the nation's first technological universities. A highly selective private university located within an hour of Boston, WPI is consistently ranked among the top 60 research institutions by US News & World Report. The university is home to an innovative and intensive project-based curriculum that empowers students with the knowledge and skills to address real world problems around the globe, an approach repeatedly cited for excellence by The Fiske Guide to Colleges and The Princeton Review.

Located in the heart of New England, WPI is surrounded by cultural and recreational opportunities. The UMass Medical Center, a large number of technology companies and many colleges and universities are located in the immediate area making it ideal for two-career families.

Questions about the hiring process should be sent to recruit@cs.wpi.edu. Applications should be submitted per instructions at http://apptrkr.com/394031. You will need to include detailed research and teaching statements, vitae and contact information for at least three references.

Review of applications will begin December 15, 2013 and continue until the position is filled.

To enrich education through diversity, WPI is an affirmative action, equal opportunity employer.

DOI:10.1145/2534706.2534727

Peter Winkler

Puzzled
Solutions and Sources

Last month (November 2013) we posted three tricky puzzles concerning coin flipping. Here, we offer solutions to all three. How did you do?

1. No head start. A coin is flipped repeatedly. How long must you wait, on average, before you see a particular head-tail sequence of length five? Any fixed sequence has probability $1/2^5 = 1/32$ of being observed in a given five flips, so the answer is 32, right? Not so fast. Overlapping causes problems. What *is* true is that in an infinite sequence of random flips, the average distance between one occurrence and the next of any fixed sequence is 32. But suppose the sequence is HHHHH. Such an occurrence would provide a huge "head start" (sorry) to the next occurrence. If the next flip is a tail, you must start over, but if a head, a new occurrence has already taken place. If X is the average time needed to get HHHHH starting fresh, the average of $1+X$ and 1 is 32. Solving for X yields a startlingly high 62 flips. To reduce your expected dues to $32, you must pick a sequence whose occurrence gives no head start to the next occurrence. HHHTT is one of 10 sequences with this property.

2. Decline the privilege. You and your opponent choose different length-five head-tail sequences, with the first to occur determining the winner. If possible, decline the "privilege" of picking first; whatever the first chooser chooses, the second can do much better. Worst case: you pick HHHHH, your opponent picks THHHH, and you are dead meat unless the flips begin with five heads. In general, if you pick VWXYZ your opponent crushes you by picking UVWXY, with the right choice of U. If my calculations are correct, you can do no better than, say, HHTHT (one of the good choices in Puzzle 1). Even then, your opponent counters with HHHTH, winning two times out of three. Does it bother you that your opponent has a big advantage, even though the average waiting time for your sequence is less?

3. A good bet. You had to decide whether to risk $1 to win $19 by flipping exactly 50 heads in 100 flips. How likely is flipping 50 heads on the nose? A good estimate for flipping precisely n heads in $2n$ flips is $1/\sqrt{(\pi n/2)}$. Sheesh. What does π have to do with it? Anyway, with our modest numbers, a computer does the exact calculation easily; the number of ways to flip exactly 50 heads is "100 choose 50," or $100!/(50!)^2$, so the total number of outcomes for 100 flips is 2^{100}. Dividing the first by the second yields approximately 8%, which exceeds 1/20, so it is indeed a good bet—assuming it was not made with your bottom dollar.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science, at Dartmouth College, Hanover, NH.

PHOTOGRAPH BY ALEX KALMBACH

[CONTINUED FROM P. 128] new operating system concepts, and was implemented in a high-level programming language rather than assembly code.

What prompted the move to SRI? I gather Bell Labs stopped supporting Multics shortly before you left; was that what inspired you to search for a new position?

Actually, my mother had died in early 1970 (my father nine years before), and I suddenly had some new mobility. Also, the corporate culture of Bell Labs was beginning to change, although the UNICS/UNIX work of Ken Thompson, Dennis Ritchie, (and) Brian Kernighan certainly kept some of the old feeling of fabulous research and development. About then, Lotfi Zadeh invited me to teach at Berkeley for the academic year 1970–1971. At the time, I had three children aged 3, 5, and 8 (Fibonacci numbers, which made them easy to remember), whom I wound up raising more or less by myself for 17 years. In the spring of 1971, Jack Goldberg was the long-time head of the computer science lab at what was then Stanford Research Institute, whom I had known since David Huffman invited me to teach at Stanford in the spring of 1964. Jack invited me to consult for a week and help set up the CSL research agenda. That led to an offer to join what later became SRI International. The kids and I really enjoyed California, so we decided to stay.

Can you tell me a bit about your work on CTSRD and the clean-slate project? If I understand correctly, you have explored tagged architectures and hardware specifications that are formally verifiable. What are the latest developments?

The history of this goes back to the Provably Secure Operating System that we did at SRI beginning in 1973. That was a tagged capability clean-slate hardware-software co-design effort where every module was specified in a formal specification language that Larry Robinson and Karl Levitt created, with a formal basis evolving from Bob Boyer's work, with the architecture later really cleaned up with Rich Feiertag. The current joint projects between SRI and the University of Cambridge draw heavily on the experiences from Multics and PSOS.

“The old adage seems to prevail among folks who have not been so exposed: To a person with a hammer, everything looks like a nail.”

In the 1970s, formal methods were impractical unless everything was formally specified (as it was in PSOS). Today, we have built the SRI formal analysis tools into the hardware specification/development tools so that we can indeed, rather straightforwardly, prove properties about the hardware, and then use the Robinson-Levitt theorem from the PSOS project to prove properties of the low-layer software that use proofs of properties about the hardware, giving much greater assurance than you can get otherwise.

You have moderated the ACM group on risk for years. Thanks to that work, and to your book on computer-related risks, it has become apparent that most technology failures follow one of a small, set number of patterns. Do you have a favorite failure?

Actually, I have many favorites. What is fascinating is the huge diversity of causes and effects. Causes are often attributable to a combination of problems relating to inadequate requirements, inaccurate specifications, flawed system architectures, weak hardware, poor software engineering practices, human foibles, and malicious actions (for example). Effects span all fields of endeavor. So, I really do not think there are just a few patterns.

You are known for advocating a holistic view of computers and taking, for instance, human error and human behavior in general into account. Are people growing more receptive to this message?

Yes and no. People who have been reading the Risks Forum are repeatedly exposed to that holistic view, partly by my editorial comments and partly by the intrinsic nature of the ongoing discussions of what demonstrably turn out to be holistic problems. On the other hand, the old adage seems to prevail among folks who have not been so exposed: To a person with a hammer, everything looks like a nail. That adage applies equally to other artifacts and other disciplines. Examples include people who see everything as a hardware problem, or a software problem, or an algorithm problem, or a network problem, or a security problem, or a human safety problem. The holistic view is that those concepts are often interrelated. One of my very favorite quotes is one that Butler Lampson attributes to Roger Needham and Roger attributed to Butler: If you believe that cryptography is the answer to your problem, then you do not understand cryptography, and you do not understand your problem. That, of course, also applies to many other disciplines.

When you were still a student at Harvard, you discussed simplicity with Albert Einstein over breakfast, and you have mentioned that he told you he thought Brahms was “burning the midnight oil trying to be complicated.” What do *you* make of Brahms?

I love Brahms' music, because of its carefully integrated complexity, but also its harmonic and aesthetic beauty and the innovations it introduced at the time he was composing. C

Leah Hoffmann is a technology writer based in Piermont, NY.

DOI:10.1145/2534706.2534728

Leah Hoffmann

Q&A Securing the Risk

Peter G. Neumann views computers and their related issues holistically.

DESCRIBED IN ISCA'S *Information Security* magazine as a "designated holist," Peter G. Neumann has devoted most of his decades-long career to the big-picture view of computer systems, security, and risk avoidance. After earning his Ph.D. at Harvard in 1961, Neumann went to work at Bell Laboratories, where he was part of the pioneering group that developed the Multics operating system. Ten years later, he moved to SRI International, where he now, at the age of 81, continues to break new ground with clean-slate architecture research funded by the U.S. Defense Advanced Research Projects Agency (DARPA). He also still moderates the ACM Forum on Risks to the Public in the Use of Computers and Related Systems, adding commentary and insight to a seemingly endless stream of tales about human and machine-driven error.

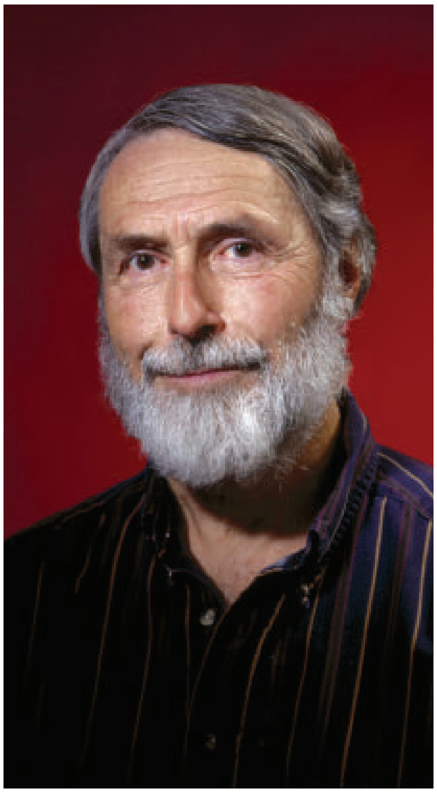


PHOTO COURTESY OF SRI INTERNATIONAL COMPUTER SCIENCE LABORATORY

"I cut my eyeteeth programming a complex matrix inversion routine on an IBM CPC (Card-Programmed Calculator)."

You studied applied math at Harvard. What drew you to computer science? Your website mentions a summer job with the Naval Ordnance Lab using an IBM Card-Programmed Calculator; were you interested in computers before that job, or did the position spark your interest?

I had always been fascinated by math in high school, and was a math major as an undergrad at Harvard. In spring 1953 of my junior year, I took Howard Aiken's course on the 'switching theory'—which was really the basis for Boolean logic applied to realistic computer hardware circuits. In that there was no computer science at that time, all of the computer-related courses during those early years were listed under 'Applied Mathematics' in the catalog. A friend of my parents suggested a 'student aid trainee' summer internship in Silver Spring, MD, so I cut my eyeteeth programming a complex matrix inversion routine on an IBM CPC, which effectively had a microcode plugboard that made it look like a three-address calculator with four registers. A foot-long card deck included the linear program (no loops) and the matrix data, which when run through the CPC punched two new cards, which when placed at the end of the card deck and run again punched out four cards. This process was then repeated until the matrix inversion was completed; very primitive, but it was a great learning experience.

You spent the first 10 years of your career at Bell Labs, where you helped lead the development of Multics. How did you come to be involved with the project?

I owe a lot to Vic Vyssotsky. On the first day of work after New Year's Day in 1965, Vic and I happened to be walking in from the parking lot together, and I asked what he was working on. He said, "Why don't you join me?" as he was just going into the very first organizational meeting for Bell Labs' participation with MIT in the as-yet-unnamed groundbreaking new computer project that today we would call clean-slate total-system architecture. I did. It, of course, involved new hardware (with virtual memory, segmentation, paging, an input-output coprocessor), [CONTINUED ON P. 127]

10th World Congress on Services (SERVICES 2014)

June 27—July 2, 2014, Anchorage, Alaska, USA

Federation of 5 Service-Centric Conferences from Different Angles

(http://www.ServicesCongress.org)

7th International Conference on Cloud Computing (CLOUD 2014)

Cloud Computing is becoming a scalable services delivery and consumption platform in the field of Services Computing. The technical foundations of Cloud Computing include Service-Oriented Architecture and Virtualizations. Major topics cover Infrastructure Cloud, Software Cloud, Application Cloud, Social Cloud, & Business Cloud. Visit http://thecloudcomputing.org.

21th International Conference on Web Services (ICWS 2014)

ICWS 2014 will feature **web-based** services modeling, design, development, publishing, discovery, composition, testing, QoS assurance, adaptation, and delivery technologies and standards. Visit http://icws.org.

11th International Conference on Services Computing (SCC 2014)

SCC 2014 will focus on **services innovation lifecycle** e.g., enterprise modeling, business consulting, solution creation, services orchestration, optimization, management, and BPM. Visit confer-

3rd International Conference on Mobile Services (MS 2014)

MS 2014 will feature **Wearable technology and applications**. Topics cover but not limited to all innovative aspects of wearable devices, programming models, integration, and domain specific solutions.

Visit themobileservices.org/2014.

International Congress on Big Data (BigData 2014)

BigData 2014 aims to explore various aspects of Big Data including modeling, storage architecture (NOSQL), enterprise transformation, text mining, social networks, applied analytics and various applications, and Big Data As A Service. Visit http://ieeebigdata.org/2014.

Conference proceedings have been EI indexed. Extended versions of invited ICWS/SCC/CLOUD/MS/BigData papers will be published in IEEE Transactions on Services Computing (TSC, SCI & EI indexed), International Journal of Web Services Research (JWSR, SCI & EI indexed), International Journal of Business Process Integration and Management (IJBPIM), IT Pro (SCI & EI Indexed), and the worldwide Services Computing community's Open Access journals (IJCC, IJBD, and IJSC).

Submission Deadlines

ICWS 2014: 1/15/2014
CLOUD 2014: 1/15/2014
SCC 2014: 1/31/2014
MS 2014: 1/31/2014
BigData 2014: 2/10/2014
SERVICES 2014: 2/10/2014

Any questions, contact the organizing committee at services.ieeecs@gmail.com

IEEE TRANSACTIONS ON SERVICES COMPUTING

IT Professional
Web Services
INTERNATIONAL JOURNAL OF WEB SERVICES RESEARCH
Business Process Integration and Management