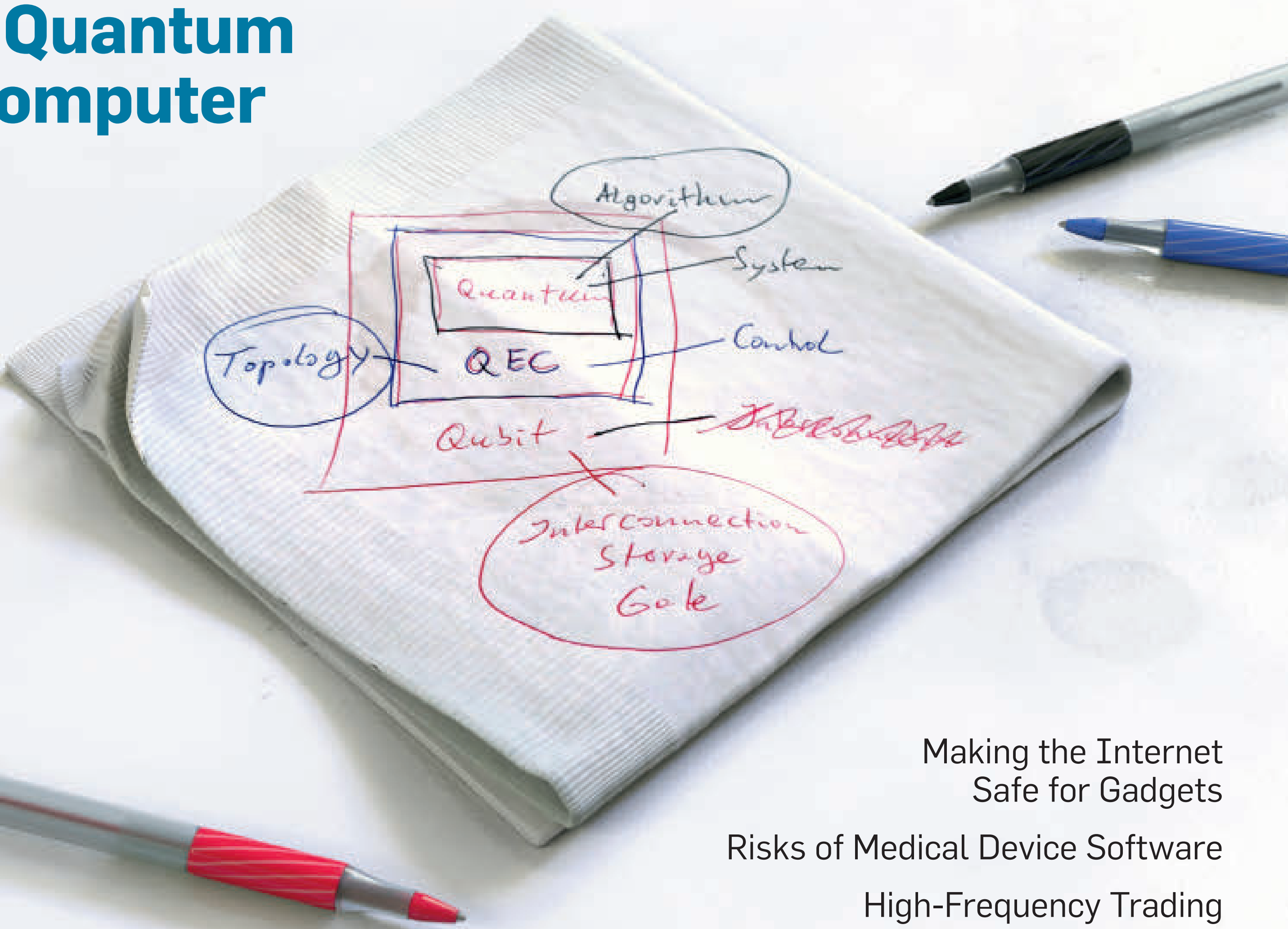
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

10/2013 VOL.56 NO.10

A Blueprint for Building a Quantum Computer



Association for Computing Machinery

VEE 2014

10th ACM SIGPLAN/SIGOPS international conference on

Virtual Execution Environments

Salt Lake City   1–2 March 2014   with ASPLOS

Authors are invited to submit original papers related to virtualization across all layers of the software stack down to the microarchitectural level

Deadline   14 November 2013

General Chair
Martin Hirzel *(IBM Research)*

Program Co-chairs
Erez Petrank *(Technion)*
Dan Tsafrir *(Technion)*

Program Committee
Remzi Arpaci-Dusseau *(UW Madison)*
David F. Bacon *(IBM Research)*
Muli Ben-Yehuda *(Technion & Stratoscale)*
Dilma Da Silva *(Qualcomm Research)*
Angela Demke Brown *(U of Toronto)*
David Dice *(Oracle)*
Ajay Gulati *(VMware)*
Sam Guyer *(Tufts U)*
Antony Hosking *(Purdue U)*
Galen Hunt *(MSR)*
Doug Lea *(SUNY at Oswego)*
Gilles Muller *(INRIA)*
Todd Mytkowicz *(MSR)*
Mathias Payer *(UC Berkeley)*
Donald Porter *(Stony Brook U)*
Karsten Schwan *(Georgia Tech)*
Liuba Shrira *(Brandeis U)*
Bjarne Steensgaard *(Microsoft)*

http://vee2014.org

in cooperation with **USENIX**

A.M. Turing Award

Call for Nominations

http://amturing.acm.org/nominate

Association for Computing Machinery

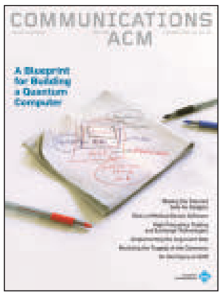
Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM

PHOTOGRAPH BY REMY MUSSER; ILLUSTRATION BY HANK OSUNA

Association for Computing Machinery
Advancing Computing as a Science & Profession

10/2013
VOL. 56 NO. 10



About the Cover:
The ultimate architecture of a quantum computer will be key to its performance, say the authors of this month's cover story (p. 84). Rodney Van Meter and Clare Horsman examine the components that should be integrated to create a blueprint for a computer with the size, and speed, and accuracy like no other. Cover illustration by Coherent Images.

ILLUSTRATIONS BY SPOOKY POOKA AT DEBUT ART, CARSTEN REISINGER, COHERENT IMAGES

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.

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

“The ecosystem that supports scholarly and professional societies is undergoing rapid change, creating significant challenges.”

DOI:10.1145/2507771.2507772 John White

On the Future of ACM

As a non-profit educational and scientific society in the computing field, ACM strives to deliver resources that advance computing as a science and a profession. Over the past decade, ACM has been enormously successful in doing so. With 109,000 members, ACM is now the world's largest professional society in computing. Its conferences and journals are among the most highly regarded in the field. Its Digital Library is world class and a major resource for the computing community. Its leadership has had an impact on computing education, diversity, and public policy.

Like most professional societies, ACM has financed this work through three primary revenue streams: membership, conferences, and publications. ACM has been successful here also, enjoying a very solid financial base. Nevertheless, the ecosystem that supports scholarly and professional societies is undergoing rapid change, creating significant challenges. These challenges include: the need for an international presence; the expectation of quality services by its members and other stakeholders; membership retention and growth; adapting to rapid changes in online technology and content delivery; and accommodating new models of publication.

For ACM, we have been growing our international presence, but significant areas have yet to be addressed. While new services are being provided to members, it has been a challenge to keep pace with changes in technology (for example, mobile devices, video services, among many others). ACM membership is growing, but SIG membership has declined and hovers around 40% of ACM membership. The current role of conferences in computing research (a main SIG activity) is being questioned, and the open access movement is changing community expectations regarding how publications should be financed and distributed. At the same time, technology developments are leading to new visions of how researchers and professionals interact: social media, virtual conferences, and "open research" in which artifacts such as data and software are archived and exchanged. What seems to be emerging is a richly interconnected world, filled with services, where everything is expected to be free.

Given this picture, it is not clear that ACM's traditional products and services, as well as the current models and means for financing them, will provide the necessary resources to address these looming challenges. As a result, senior ACM leaders and staff will be holding a two-day retreat this November to take a long, hard look at the association. At this retreat we will: unpack and understand ACM's current business model(s); understand the challenges and forces at play for each model; assess the strengths/weaknesses of each model relative to emerging challenges and risks; and explore new thinking and new business models for the richly interconnected set of current and future ACM activities, programs, and products. The main goal of the retreat is to answer the question:

How must ACM restructure its portfolio, and the business models that support it, to stay relevant and viable in the future?

If you have thoughts on the future of ACM given the challenges we face, I would like to hear from you.

John White (CEO@acm.org) is the chief executive officer of ACM.

LEARNING @ SCALE

ATLANTA, GEORGIA MARCH 4–5 2014

Call for Papers

LEARNING @ SCALE

ACM will host the first **ACM Conference on Learning at Scale** to be held March 4-5, 2014 in Atlanta, GA.

Inspired by the emergence of Massive Open Online Courses (MOOCs) and the shift in thinking about education, ACM created this conference as a new scholarly venue to explore how learning and teaching can improve when done "at scale."

ABOUT THE CONFERENCE

Learning at Scale refers to new approaches for students to learn and teachers to teach, when engaging hundreds or even thousands of students; be it face-to-face or remotely, synchronous or asychronous.

Topics will include (but are not limited to) Usability Studies, Tools for Automated Feedback and Grading, Learning Analytics, Investigation of Student Behavior and Correlation with Learning Outcomes, New Learning Tools and Techniques at Scale.

CALL FOR PAPERS

Authors are encouraged to visit the Learning @ Scale website **http://learningatscale.acm.org/**

- ➤ for details about the conference and guidelines for paper submissions, including an array of subject examples to explore.
- ➤ Authors are encouraged to consider submitting a full paper (10-pages max); a tutorial proposal for a 1-4 hour discussion on a relevant tool, technology, or methodology related to learning at scale; or a work-in-progress poster or demo.
- ➤ Papers must tackle topics "at scale."

IMPORTANT DATES

NOVEMBER 8, 2013
Paper submissions due.

NOVEMBER 8, 2013
Tutorial proposals due.

DECEMBER 23, 2013
Notification to authors of accepted full papers.

JANUARY 2, 2014
Work-in-progress submissions due.

JANUARY 14, 2014
Notification to authors of accepted work-in-progress.

JANUARY 17, 2014
ALL revised and camera-ready material due.

DOI:10.1145/2507771.2507773

Vinton G. Cerf

Revisiting the Tragedy of the Commons

In 1968, Garrett Hardin published an essay in *Science* magazine[a] entitled "The Tragedy of the Commons" in which he focused on indiscriminate population growth as an issue.

The concept of the *commons* and its overuse was not then new and had been applied to the overgrazing of common or public properties made available, for example, to cattle and sheep owners. The primary message tended to be that the limitation of resources should be grounds for regulated access through either law or custom. In 2012, Bill Davidow applied that concept to the Internet[b] albeit in a prophetic way, since he was worried about the loss of privacy in the Internet commons.

This column is based on the view that the resources of the Internet, while finite, are not bounded except by our ability to construct more resource to grow the shared virtual space the Internet and its applications create. That the characteristic parameters of the Internet have increased by a factor of over one million since the system was activated in January 1983 is testament to the feasibility of growing Internet capacity to meet demands and need. The investments required to achieve this growth have come from many sources but largely from the private sector. What is important to appreciate is that private sector investments have been made voluntarily for business or other reasons and not under coercion or even government mandate. This is the remarkable property of the Internet: its protocols and implementation are a consequence of bottom-up, collaborative processes and independent decision making by the implementers and operators of the autonomous systems that make up the Internet.

A consequence of these features is the Internet escapes the usual tragedy scenario by responding to demand with increased capacity. This is not to say the Internet is without overuse. It can congest with the manifestation that applications are slow or do not work at all owing to delays that trigger timeouts. Unlike physically limited space for grazing sheep, however, the Internet's capacity can be expanded if there is willingness and ability to invest.

There is, however, another way in which the Internet does resemble a commons. It is a shared, virtual environment, derived from physical implementation using routers, transmission technology (wired, wireless) and host computers (including gigantic data centers and individual laptops, tablets, and mobiles) at the "edge" of the Internet. The users of the Internet share a common environment that permits them to exchange information, to access resources, and to carry out computations and, more generally, applications. A consequence of this sharing is that we may experience common risks associated with malefactors seeking to harm others, to disrupt communication, and to commit a variety of abuses. These abuses manifest in many ways, such as fraud, theft, misrepresentation, disruptive malware, and hijacking of resources, to name just a few examples.

Whether these abuses are violations of law will vary from one jurisdiction to another, particularly across international boundaries. Coping with these problems is not a trivial matter and may require cooperation across jurisdictions, instantiation of international treaties, formation of informal working groups, R&D leading to more robust operating systems and Web-based tools and applications, and even changes in social behavior. These problems are made all the more difficult to cope with because of the asymmetry between perpetrators of abuse and their victims. Small groups can inflict a great deal of damage that might have required nation-state level resources in the past but, thanks to rapidly evolving computing power, these resources may be available to individuals.

The co-opting of computers owned or operated by businesses, government, and the public to form so-called "botnets" used to inflict damage, generate spam, or to launch denial-of-service attacks is made possible by the exploitation of, inter alia, operating system or application software bugs, lax security procedures, and insider cooperation. Some of these problems lie squarely in the professional spaces of ACM members and thus pose a challenge to us as members or as practitioners of software and hardware engineering. They are certainly subject to serious research initiatives aimed at their mitigation.

The conclusion, then, is while the commons created by the Internet need not be bounded, it is a shared environment that must be protected for the benefit of its users. If there is a potential tragedy here, it will be that the Internet becomes too unsafe for reliable use. It seems to me of vital importance to fend off such an outcome so the benefits of this commons can be realized by all of humanity.

Vinton G. Cerf, ACM PRESIDENT

a G. Hardin. "The Tragedy of the Commons." *Science 162*, 3859 (Dec. 1968), 1243–1248; doi:10.1126/science.162.3859.1243.

b B. Davidow. "The Tragedy of the Internet Commons." *The Atlantic* (May 18, 2012); http://bit.ly/KqGN0x/

DOI:10.1145/2507771.2507774

Deep Accountability, Beyond Even Liability

I AM A Colorado licensed professional engineer whose area of practice is software and who found no cause for disagreement with the first half of Vinton G. Cerf's "From the President" editorial "But Officer, I Was Only Programming at 100 Lines Per Hour!" (July 2013). The second half was another matter.

I concur with Cerf's statement: "I think many of you would agree that a test or questionnaire is not likely to provide assurance that a 'certified professional' programmer's work is free of flaws..." to the delight of my liability insurance provider, who repeatedly admonishes me to avoid giving guarantees or warranties against errors in my work. But "free of flaws" is an unrealistic expectation for any human being, even a licensed professional software engineer. What my professional license (together with the PE exam and years of education and decades of professional practice behind it) does provide the public is assurance that my work is relatively free of sophomoric programming errors like buffer overflows, opportunities for SQL injection, memory leaks, and race conditions that infest the products of the more junior software developers I often must clean up after. Yes, I still make mistakes, but I make them more rarely and at a much higher level than the typical "code monkey."

Cerf also said "...it is very tempting to imagine a balance between certification and liability is worth some consideration." But if he were a patient on a treatment table under the aimpoint of a Therac-25 [radiation therapy machine], where would he prefer that balance to be placed?

And "I take pride in believing that I, and many colleagues, see themselves as professional in spirit if not in name, and that we strive to deliver reliable code." I take the same pride, but I must also contend with the "It ran okay once, so ship it!" mentality of the typical "Let the free market decide"-oriented manager.

Moreover, "So accepting liability that the code won't break or be broken is a pretty scary thought." Okay, what if this entire discussion were about bridges and structural engineers, not code and software engineers? Professional bridge designers somehow manage to perform their work competently, accepting possible liability without living their lives in constant fright. They do so by virtue of careful design through proven principles, incorporating lots of margin into their construction, and performing rigorous testing. So it can and should be with professional software engineers.

And finally "We are so dependent on an increasing number of programs, large and small, it is difficult to believe the software profession will escape some kind of deep accountability in the future." This is one of the best endorsements for software engineering licensure I have ever seen.

Lawrence Stalla, Colorado Springs, CO

Vinton G. Cerf's view (July 2013) of "registered" and "professional" designations (and lack of standardization) among U.S. software engineers made me wonder if a model could not be borrowed from an overseas friend, the British Computer Society (http://www.bcs.org/). In the U.K., industry can often ignore BCS qualifications, but where safety concerns require a "responsible engineer," suitably qualified, it is the BCS that sets the bar. The U.K. government has delegated issuing registered- and professional-type designations to bodies like BCS that belong to the U.K.'s Engineering Council (http://www.engc.org.uk/). Though this practice may represent a vestige of a medieval guild system, "professionalism" in these terms would be judged instead by a jury of one's peers. Yes, there is indeed the possibility of elitism (setting the bar high to exclude newcomers), but, at least where safety is concerned, better too high than too low.

Without knowing the details of each U.S. state's program, it is impossible to assess the complexity of establishing a single standard, but in the 1990s accreditation by the BCS required a minimum level of work experience, passing a written exam (or exemption for certain college degrees), providing documentation to demonstrate experience and career progression beyond "entry level," and submitting to a panel interview.

It does not seem unrealistic for ACM to help define minimum education and experience criteria for a "responsible engineer" accreditation in IT; ACM is after all an organization of IT experts and practitioners, many of whom (I would hope) to whom it would apply. ACM could administer the process itself, with lobbying to have each state recognize it (perhaps eventually a de facto national, or even global, standard); alternatively, it could be provided more simply as recommended guidelines.

Stephen Garriga, Blairsville, GA

Vinton G. Cerf (July 2013) discussed professional licensure, a controversial topic in the computing community. While our aim here is not to explore the larger question of licensure, we would like to suggest the community would benefit from practices that promote professional awareness and self-accountability, the way the Hippocratic Oath promotes self-accountability among medical doctors beginning their careers in service to society. Engineers in the U.S. have long used an obligation ceremony sponsored by the Order of the Engineer (http://www.order-of-the-engineer.org/) in which graduating seniors pledge to uphold the standards and dignity of the profession through an oath including parts of the Canon of Ethics of major international engineering societies. The oath and the ceremony are not tied to the licensure of engineers in any way but are meant to inspire young professionals toward a consciousness of the profession and remind older professionals of their responsibility receiving, welcoming, and supporting them.

An organization called The Pledge of the Computing Professional (http://

www.computing-professional.org/) founded in 2010 at Ohio Northern University and the University of South Florida aims to recognize graduates of computing programs as professionals in service to society, as the Order of the Engineer does with graduates of U.S. engineering programs. Today, 26 U.S.-based institutions conduct The Pledge rite-of-passage ceremony as part of their graduation activities. Graduates taking The Pledge sign a certificate both publicly and in the presence of their peers and are then presented a pin (http://www.computing-professional.org/symbol.html) that serves as an additional reminder of their commitment to self-accountability through ethical and moral behavior within the profession.

The Pledge has been endorsed by the Order of the Engineer, the ACM Special Interest Group on Computers and Society (http://www.sigcas.org/), and the ACM Committee on Professional Ethics. For more, please see http://computing-professional.org/ or contact any of us directly.

Ken Christensen, Tampa, FL
John K. Estell, Ada, OH
Ben Kuperman, Oberlin, OH

Author's Response:

I appreciate very much these readers' thoughtful comments and the points they make. Some of the best programmers I have known have lacked formal training, having come by their expertise through field experience. However, those points persuade me an effort might be made to say something about minimum curriculum and performance, if not an actual examination like the bar exam for lawyers or board certification for medical professionals. Evidence of continued education might also give credibility to the credentials of a professional software engineer. While it may not be inevitable, it seems sufficiently plausible that some kind of licensing will be proposed, in which case ACM might contribute positively to the development of credible course content and perhaps also testing standards for professional software engineers.

Vinton G. Cerf, ACM President

What Evidence of IT Acceleration?

In his Viewpoint "Could Artificial Intelligence Create an Unemployment Crisis?" (July 2013), Martin Ford repeatedly assumed "Information technology will continue to accelerate…" But nothing accelerates indefinitely, and many technologies, including transportation and space flight, have not seen accelerated progress in decades.[1] It is quite possible that progress in information technology will likewise reach a plateau of incremental improvement. Ford's apocalyptic vision may come about but should not be based on assumptions for which there is no evidence.

Moti Ben-Ari, Rehovot, Israel

Reference

1. Ben-Ari, M. The end of science revisited: The case for incrementalism in the future of science. *Skeptic Magazine 13*, 2 (2007), 20–27.

Author's Response:

As Ben-Ari says, acceleration does eventually slow down and it is reasonable to assume IT will likewise experience such deceleration. However, it is also not likely to occur soon. Even if advances in hardware (per Moore's Law) were to plateau, progress could continue to accelerate along other fronts (such as software performance, parallel computing, and new architectural breakthroughs). Even if the doubling period for IT acceleration would lengthen significantly, it would still imply rapid progress in light of performance levels already achieved.

Martin Ford, Sunnyvale, CA

Free Ismail Cem Bakir

As former vice-chair of the ACM Committee on Scientific Freedom and Human Rights, I would like to call your attention to the violation of human rights of Ismail Cem Bakir, a student at Istanbul Technical University in Turkey. The Committee of Concerned Scientists (http://concernedscientists.org/) (I am vice-chair, computer science) is an independent organization of scientists, physicians, engineers, and scholars devoted to the protection and advancement of human rights and scientific freedom for colleagues worldwide. We are concerned for Bakir, as well as for other students arrested during antigovernment demonstrations in Turkey in June.

On July 16, *The New York Times* reported (http://nyti.ms/14WJUt7) the Istanbul Bar Association said the police, citing terrorism laws, had issued a temporary order withholding legal assistance to these detainees, as well as access to their families. Bakir, who was studying computer engineering, was arrested in July. His sister said the police confiscated computers, books, and magazines and even threatened to charge the lawyer assisting his family. This set of events seems to confirm the Bar Association's complaint concerning the police.

Denial of access to counsel is expressly prohibited by the Universal Declaration of Human Rights and the International Covenant on Civil and Political Rights, to which Turkey is a signatory, as well as by the European Convention on Human Rights, to which Turkey is a party. Bakir's right to peaceful protest is also protected by the Covenant.

The Committee of Concerned Scientists has urged the Turkish government to investigate Bakir's arrest and detention, as well as that of others denied legal assistance and access to their families. It has also urged the government to arrange Bakir's immediate release (if he is still being held in jail) on the basis of his right to peaceful protest and expression of opinion, allowing him to resume his studies.

I urge you to help advocate Bakir's scientific freedom and human rights. Please send letters of support to

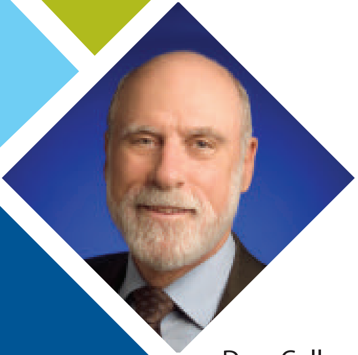
His Excellency Abdullah Gül
President
Office of the President
Cumhurbaşkanlığı
06689
Çankaya, Ankara,
Republic of Turkey

and

His Excellency
Recep Tayyip Erdogan
Prime Minister
Office of the Prime Minister
Basbakanlik 06573
Ankara, Republic of Turkey

Jack Minker, College Park, MD

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.



Association for Computing Machinery

Global Reach for Global Opportunities in Computing

Dear Colleague,

Today's computing professionals are at the forefront of the technologies that drive innovation across diverse disciplines and international boundaries with increasing speed. In this environment, ACM offers advantages to computing researchers, practitioners, educators and students who are committed to self-improvement and success in their chosen fields.

ACM members benefit from a broad spectrum of state-of-the-art resources. From Special Interest Group conferences to world-class publications and peer-reviewed journals, from online lifelong learning resources to mentoring opportunities, from recognition programs to leadership opportunities, ACM helps computing professionals stay connected with academic research, emerging trends, and the technology trailblazers who are leading the way. These benefits include:

Timely access to relevant information

- ***Communications of the ACM*** magazine
- ***ACM Queue*** website for practitioners
- Option to subscribe to the ***ACM Digital Library***
- ACM's ***50+ journals and magazines*** at member-only rates
- ***TechNews***, tri-weekly email digest
- ***ACM SIG conference*** proceedings and discounts

Resources to enhance your career

- **ACM Tech Packs**, exclusive annotated reading lists compiled by experts
- **Learning Center** books, courses, webinars and resources for lifelong learning
- Option to join **36 Special Interest Groups** (SIGs) and **hundreds of local chapters**
- **ACM Career & Job Center** for career-enhancing benefits
- ***CareerNews***, email digest
- **Recognition of achievement** through Fellows and Distinguished Member Programs

As an ACM member, you gain access to ACM's worldwide network of more than 100,000 members from nearly 200 countries. ACM's global reach includes councils in Europe, India, and China to expand high-quality member activities and initiatives. By participating in ACM's multi-faceted global resources, you have the opportunity to develop friendships and relationships with colleagues and mentors that can advance your knowledge and skills in unforeseen ways.

ACM welcomes computing professionals and students from all backgrounds, interests, and pursuits. Please take a moment to consider the value of an ACM membership for your career and for your future in the dynamic computing profession.

Sincerely,

[illegible]

Vint Cerf

President
Association for Computing Machinery

acm Association for Computing Machinery

Advancing Computing as a Science & Profession

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

BLOG@CACM

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:10.1145/2507771.2507775 http://cacm.acm.org/blogs/blog-cacm

Helping Scientists, Engineers to Work Up to 100 Times Faster

Philip Guo explains how programming skills can make scientists and engineers more efficient and creative.

Philip Guo
Why Scientists and Engineers Must Learn Programming
http://cacm.acm.org/blogs/blog-cacm/166115-why-scientists-and-engineers-must-learn-programming/fulltext
July 18, 2013

In recent years, there has been an admirable push to get more people to learn programming. But if I have never been exposed to programming, why should I invest all of the effort to learn? What is in it for me?

Pundits often give fuzzy responses, like claiming that programming is the "literacy of the 21st century," that it helps you become a more empowered citizen, and that it enables you to create magical works of pure creativity.

Even though I agree with many of those claims, I am not convinced that they are concrete enough to motivate someone to devote the thousands of hours necessary to get proficient at programming.

Instead of trying to convince everyone to learn programming, I have a more modest goal: encouraging scientists and engineers. Here is my value proposition to them:

If you are a scientist or engineer, programming can enable you to work 10 to 100 times faster, and to come up with more creative solutions than colleagues who do not know how to program.

Kevin's Story

Modern-day scientists and engineers are spending more and more of their workdays in front of the computer. As an example, consider my friend Kevin, who works in oceanography and mechanical engineering. Whoa, sounds like he is probably spending all day out on high-tech boats, rigging together mechanical devices like MacGyver and collecting data from underwater sensors, right? This must be his typical workday — hard hats and heavy-duty work gloves.

Actually, Kevin spends less than 5% of his time out on the ocean; the other 95% of the time, he is sitting in front of the computer writing programs to clean up, transform, process, and extract insights from data collected out in the field.

The same story plays out for scientists and engineers in all sorts of fields: astronomers, biologists, physicists, aerospace engineers, economists, geneticists, ecologists, environmental engineers, neuroscientists...the list goes on and on. Modern-day science and engineering is all about processing, analyzing, and extracting insights from data.

Three Reasons to Learn

Over the past few years, many scientists and engineers have ranted to me about how furious they are that nobody made them learn programming back in high school or college. They now realize how much more productive they could be at work if they had developed those skills earlier.

Based on these conversations, I've come up with three reasons why scientists and engineers must learn programming:

1. You can work 10 times faster by writing computer programs to automate tedious tasks (such as data cleaning and integration) that you would otherwise need to do by hand. If you know how to program, computer-related tasks that used to take you a week to finish will now take only a few hours. I cannot think of any other skill that leads to an instant 10-times productivity boost for scientists and engineers.

2. Programming allows you to discover more creative solutions than your colleagues who do not know how to program. It lets you go beyond simply using the tools and datasets that everyone else around you uses, to transcend the limitations that your peers are stuck with. For example, you will be able to write programs to automati-

cally acquire data from new sources; to clean, reformat, and integrate that data with your existing data, and to implement far more sophisticated analyses than your colleagues who can only use pre-existing tools. By doing so, you are more likely to make a creative innovation that your colleagues would not even think of exploring due to lack of programming skill.

3. Finally, knowing how to program allows you to communicate effectively with programmers that your lab hires to do the heavy-duty coding. I do not expect you to become as adept as the professionals, but the more you know about programming, the more you will be able to relate to them and to command their respect. If you can motivate programmers in your lab to spend more of their time helping you solve technical problems (e.g., by writing parallel programs that run on a compute cluster), you can work 100 times faster than if you had to attack those problems alone.

Thoughts?

Email philip@pgbovine.net

Postscript

Readers have responded with two main classes of comments:

1. Scientists and engineers in lots of fields already learn some amount of programming (e.g., in Excel, MATLAB, Mathematica, LabVIEW).
2. We should strive to create end-user programming tools that make it easy enough for scientists and engineers to do what they need without even knowing that they are programming.

I agree with both of these points. But in the foreseeable future, I think that programming skill will always be positively correlated with creative productivity in many technical fields. Thus, for scientists and engineers who already know some amount of programming, learning more will always provide a competitive advantage over colleagues who are not as adept. We might someday get to a future where programming as we know it will become as obsolete as calculating integrals by hand, but I doubt that is going to happen anytime soon.

Readers' Comments

FWIW, my research credo (as you may know, Philip) is that we have to go to them, not the other way around. Programming is always a good skill to have, but asking people with immense amounts of domain knowledge (that took years to acquire) to ***also*** *be proficient coders (another skill it takes a lot of time to learn to be competent at) is simply not feasible. Ideally, we should get to a place where the UIs and tools make it so they do not even know they are programming.*

Perhaps put another way, I do not believe there is something special about these domains that requires the ability to do general-purpose programming when many other domains have been having to manipulate data for a much longer time and yet do not require programming skills. We have managed to create special-purpose tools that are narrow enough to be relatively easy to use but broad enough to cover a wide number of problems. Think Excel, for instance. Even professions like financial analysts, at worst, have to contend with SQL. And even that, I would argue, is asking too much.

All that said, perhaps we would agree that, sure, maybe they do ultimately need to program, but there is still a huge mismatch between the programming tools they currently have and what they need. I will agree they need to learn to "program" if you agree we potentially need to change what it means for them to "program." ;)

—Nicholas Murphy

Hi Nick,

I am totally with you. I would love to get to a world where end-user programming tools for scientists are powerful and usable enough that they can do the kinds of sophisticated analyses that they need without even thinking that they are programming. But as it stands, the amount that can be accomplished by existing tools is fairly limited; thus, as a scientist, having some understanding of programming will always give you a comparative advantage vs. your peers who can use only, say, Excel. When programming skills no longer become a comparative advantage in science, that is one sign that the tools have gotten powerful enough. Maybe an analogy is that nowadays, being able to do surface integrals or geometric proofs or draw pretty scatterplots on paper no longer provide a comparative advantage, since computer-based tools make it easy enough to do so.

—Philip Guo

Greg Wilson has been trying to define what part of computing that scientists and engineers need to be productive today. He has been evolving his Software Carpentry workshops (http://software-carpentry.org) to focus on what scientists and engineers need, not everything that computer scientists find valuable.

—Mark Guzdial

Thanks for the pointer, Mark! I am a big fan of Greg's Software Carpentry work and would love to see more efforts to specialize programming curricula for working scientists/engineers.

Many people in the CS Ed community have already done a great job at introducing programming to a variety of audiences — e.g., elementary/middle school kids, high school robotics aficionados, aspiring digital artists — and I would love to see more outreach to the scientific/engineering communities.

—Philip Guo

If I may be so bold as to suggest a fourth reason that it is required:

Programming requires you to break big problems down into their smallest discrete components, and then to solve the big problem by systematically solving those smaller components. This is also exactly the kind of thinking that is required 90% of the time in science and engineering. Once programming has forced you to learn how to think this way, it is far easier to apply it to non-programming problems.*

*in the 90/10 perspiration vs. inspiration breakdown

—Anonymous

I am reading Phil's statement and I'm skeptical about reason 3, which claims one could work 100 times faster using parallel computers. The computer may run 100 times faster, but that will not save me 99% of my time. Usually, computing cycles are not a good measure of productivity. On the other hand, I do not think we need "thousands of hours" to become a programmer. I think it is in the "hundreds of hours." A typical course at a university is less than 40 hours of instruction, and less than 120 hours of assignments. Five programming courses, 5 x 160 = 800 hours, is enough. That is one semester. That is enough to use MATLAB/Maple/ Mathematica/ C++ proficiently.

—Michael Monagan

Philip Guo is a postdoctoral scholar in the Massachussetts Institute of Technology Computer Science and Artificial Intelligence Laboratory.

Call for Nominations

The ACM Doctoral Dissertation Competition

Rules of the Competition
ACM established the Doctoral Dissertation Award program to recognize and encourage superior research and writing by doctoral candidates in computer science and engineering. These awards are presented annually at the ACM Awards Banquet.

Submissions
Nominations are limited to one per university or college, from any country, unless more than 10 Ph.D.'s are granted in one year, in which case two may be nominated.

Eligibility
Please see our website for exact eligibility rules. Only English language versions will be accepted. Please send a copy of the thesis in PDF format to emily.eng@acm.org.

Sponsorship
Each nomination shall be forwarded by the thesis advisor and must include the endorsement of the department head. A one-page summary of the significance of the dissertation written by the advisor must accompany the transmittal.

Deadline
Submissions must be received by **October 31, 2013** to qualify for consideration.

Publication Rights
Each nomination must be accompanied by an assignment to ACM by the author of exclusive publication rights. (Copyright reverts to author if not selected for publication.)

Publication
Winning dissertations will be published by ACM in the ACM Digital Library.

Selection Procedure
Dissertations will be reviewed for technical depth and significance of the research contribution, potential impact on theory and practice, and quality of presentation. A committee of individuals serving staggered five-year terms performs an initial screening to generate a short list, followed by an in-depth evaluation to determine the winning dissertation.

The selection committee will select the winning dissertation in early 2014.

Award
The Doctoral Dissertation Award is accompanied by a prize of $20,000 and the Honorable Mention Award is accompanied by a prize of $10,000. Financial sponsorship of the award is provided by Google.

For Submission Procedure
See http://awards.acm.org/doctoral_dissertation/

acm Association for Computing Machinery

Science | DOI:10.1145/2507771.2507776

Alex Wright

Tuning In to Graphene

New ultra-fast wireless antennas may be on the way, but don't throw away your old wireless router just yet.

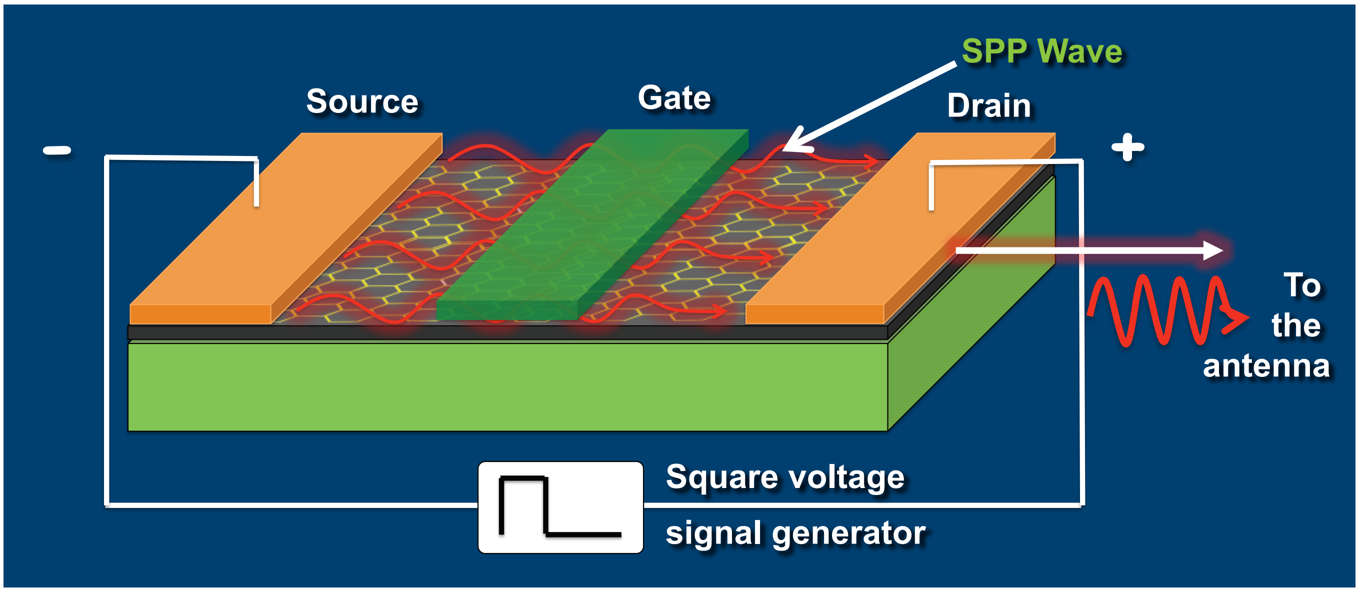
In March of this year, a team at the Georgia Institute of Technology made headlines when it revealed plans for a new microscopic antenna built out of graphene, a synthetic form of carbon with remarkable conductive properties.

Early press coverage focused on the promise of speedier wireless connections, and with good reason: such an antenna could, in principle, allow for terabit-per-second transfer speeds—fast enough to download a high-definition movie in a fraction of a second. At distances of a few centimeters, download speeds could approach an astonishing 100 terabits per second—the equivalent of three months' worth of HD footage.

While the prospect of faster downloads might come as welcome news to legions of everyday Web surfers, the technology's long-term potential extends well beyond the problem of easing Internet bottlenecks. Researchers are starting to explore a wide range of potential applications for graphene nano-antennas, such as linking the internal components of electronic devices or creating fine-tuned sensor networks capable of thwarting chemical or biological attacks.

Before any of those ideas can come to fruition, however, researchers will need to overcome a number of formidable hurdles—not least of which involves the prohibitive cost of making graphene in the first place.

In 2004, Andre Geim and Konstantin Novoselov of the University of Manchester invented graphene by taking strips of graphite (the same technology that pow-



The structure of a graphene-based plasmonic nano-transceiver for wireless communication in the terahertz band.

IMAGE COURTESY OF IAN AKYILDIZ/GEORGIA TECH ers the humble Number 2 pencil) and attaching them to adhesive tape, then repeatedly separating the strips until they had isolated a one-atom thick layer of the substance they dubbed graphene.

Ever since that discovery—which earned Geim and Novoselov a Nobel Prize in 2011—graphene has generated a level of excitement in the materials engineering world not seen since the late-1990s boom in carbon nanotubes.

Some might argue that carbon nanotubes offer a cautionary tale: a much-hyped new material that failed to live up to its initial promise. Graphene's real-world prospects remain uncertain, given its current manufacturing cost of $3,000 per square meter, but its promoters hold out hope that researchers will address that challenge in the years to come.

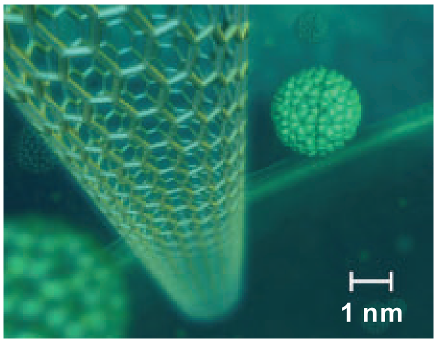
Several major university research initiatives are now under way to explore new techniques for working with graphene, some of them supported by a major research funding initiative from the European Union. Meanwhile, companies like IBM, Intel, and Samsung are moving quickly to perfect new techniques for synthesizing the material.

This flurry of activity stems from graphene's seemingly incredible promise. It is exactly one atom thick—as close to a two-dimensional material as possible. It is also nearly transparent, and flexible enough to take almost any form, yet it is also extremely dense—harder than steel or diamond.

That remarkable combination of lightness, flexibility, and strength has prompted scientists to envision any number of potential manufacturing uses, including ocean liners, airplanes, automobiles, washing machines, transparent touch screens, and new kinds of solar cells.

Beyond its remarkable structural properties, graphene also boasts a perfect honeycomb structure that allows electrons to move with almost no resistance, up to 500 times faster than in silicon. That level of conductivity makes it ideally suited for building transistors, processors, memory, cellphones—or antennas.

Like carbon nanotubes, graphene features plasmons (electron oscillations) in one-dimensional structures that support the transmission of surface waves at frequencies in the terahertz band (0.1THz–10THz). Many other materials, like fibers and plastics, are transparent at these frequencies, but graphene's dense structure allows it to propagate these signals quite effectively.



Nanomaterials enable new technologies.

The Georgia Tech team, led by Ian Akyildiz, has proposed making an antenna out of graphene by creating pieces of graphene that are between 2 and 100 nanometers wide and one micrometer long, each capable of detecting electromagnetic waves in the terahertz frequency band. By using ultra-thin graphene nanoribbons instead of relatively larger graphene sheets, the team has found they can enhance the propagation of surface plasmon polariton waves, thus improving performance. The team's results will appear in a forthcoming issue of IEEE's *Journal on Selected Areas in Communication*.

Such antennas would be small enough to fit inside any number of microelectronic components. By stitching together multiple components in this way, the nano-antennas could create an entirely new kind of nanonetwork in which devices communicate directly with each other.

Graphene is just one atom thick, nearly transparent, and flexible enough to take almost any form, yet it is also extremely dense—harder than steel or diamond.

In addition to providing a way to link all kinds of devices directly to each other and to the Internet, commercial graphene nano-antennas could also enable new kinds of access networks for 5G systems and small cell-enabled cellular network architectures for beyond 4G (B4G) networks.

Communicating over the terahertz band would also allow for higher bandwidth than conventional microwave or gigahertz bands. Those performance gains come at a price, however; graphene nano-antennas work best at ranges of less than one meter.

"The advantage of the terahertz band is the gigantic capacity," says Akyildiz. "The disadvantage is that they are distance-limited, because it is so fine-grained that the water particles and gas molecules in the air affect the signal propagation. Gas molecules are the biggest enemy."

Given these limitations, most research is currently focused on applications that involve communicating over short distances. That line of research has been evolving for several years, since well before the invention of graphene.

Peter J. Burke of the University of California, Irvine, conducted some of the foundational work on nanoantennas, developing the first RF circuit model for carbon nanotubes. Based on that early work, his team began to contemplate the possibility of nanoscale antennas, developing the first theoretical models of a carbon nanotube antenna, predicting—correctly—that they would work well at terahertz frequencies.

Burke's team has since moved on to working with graphene, in part because graphene holds a major advantage over carbon nanotubes: the ability to tune its conducting properties.

"Graphene is interesting mainly because it enables very efficient dynamic control of the antenna's parameter at terahertz frequencies," says Julien Perruisseau-Carrier of the École Polytechnique Fédérale de Lausanne in Switzerland, whose team performed important early work on graphene an-

IMAGE COURTESY OF IAN AKYILDIZ/GEORGIA TECH tennas, in particular analyzing the possibilities of dynamic reconfiguration across a wide range of frequencies.

Such reconfigurability could allow graphene antennas to save power and limit interference, as well as perform highly targeted sensing.

"We are very excited because we can tune the electrical antenna properties of graphene with a DC voltage," says Burke. By changing the sheet resistance, his team has been able to lift the impedance of a graphene nanoantenna—something not possible with other kinds of nano-antennas.

In a recent paper, Burke's team proved that graphene could function over a broad frequency range—at DC, 10GHz, 100GHz, and 100GHz–1.5THz in a single sweep. The team was able to measure the graphene sheet's impedance with a novel spectrometer built by Elliott Brown at Wright State University in Ohio. Recently, the team has built on this work to tune the graphene antenna across the entire band from 100GHz to 1.5THz.

Given these physical limitations, researchers are focusing on scenarios that would benefit from placing nanoantennas and transmitters in close proximity to each other, to create what Akyildiz describes as "an Internet of nano-things."

At the most pedestrian level, such a network could offer hope of speeding up the "last mile" bottleneck that bedevils so many everyday Internet connections: the humble wireless router. For all the much-heralded advances in network speeds over the past few years—FIOS, Terabit Internet, and IPv4, to name a few—many of us have yet to see the full promise of these advances, thanks to the speed limits inherent in the 802.11n standard that most wireless routers follow.

A full-fledged nanonetwork could go much further than replacing old Wi-Fi routers, however. It could also, in theory, harvest vibrational or electromagnetic energy from the environment to reduce power consumption.

Looking further ahead, researchers are also starting to imagine long-range applications of highly miniaturized antennas. "Imagine what you could do if you could build a radio that could fit inside of a single cell?" asks Burke, whose team is now exploring the possibility of graphene nanoantennas that could bind to DNA.

Burke's team is working with Ned Seeman at New York University and Michael Norton at Marshall University, both of whom have done extensive work with DNA "origami," to explore the possibilities of nanoscale networks for DNA sensing. If successful, this initiative could bring graphene nanoantennas closer to the world of biochemistry.

As intriguing as some of these ideas may seem, the practical challenges remain daunting. At the most fundamental level, nanoantennas cannot work by themselves without additional components such as nanotransmitters and nanoreceivers—neither of which exist yet. Akyildiz's team is currently applying for patents for graphene-based nanotransmitters and nanoreceivers.

In the meantime, several other teams around the world are pursuing related ideas for graphene-based antennas and related devices. While the early results are promising, the real work is just getting under way.

"We have one idea about nano-antennas, but there are other people with other ideas about nano-antennas," says Akyildiz. "It is a race." C

Further Reading

Akyildiz, I. F. and Jornet, J. M.
The Internet of Nano-Things. *IEEE Wireless Communication Magazine*, vol. 17, no. 6, pp. 58-63, December 2010.

Burke, P. J., Li, S., and Yu, Z.
Quantitative Theory of Nanowire and Nanotube Antenna Performance. *IEEE Transactions on Nanotechnology*, 5(4), 314-334 (2006).

Perruisseau-Carrier, J.
Graphene for antenna applications: opportunities and challenges from microwaves to THz (invited). Loughborough Antennas & Propagation Conference (LAPC2012), UK, 2012.

Rouhi, N. et al.
Broadband Conductivity of Graphene from DC to THz, *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on* , vol., no., pp.1205,1207, 15-18 Aug. 2011doi: 10.1109/NANO.2011.6144485 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6144485&isnumber=6144287

Alex Wright is a writer and information architect based in Brooklyn, NY.

ACM Member News

LOVE OF SOLVING PROBLEMS SPURS YELICK ON PROGRAMMING

A love of using computational solutions to solve problems is what inspired Katherine Yelick, professor of computer science at the University of California, Berkeley (UC Berkeley), to help develop the Unified Parallel C (UPC) and Titanium programming languages. "I love designing programming languages to make computers, particularly supercomputers with parallelism, less expensive and easier to use," Yelick says.

Yelick and UC Berkeley Electrical Engineering and Computers Sciences chair David Culler co-invented UPC in 1996; the Titanium language soon followed. Yelick has demonstrated the languages' applicability across architectures using novel runtime and compilation methods. She also co-developed techniques for self-tuning numerical libraries, including the first self-tuned library for sparse matrix kernels, which automatically adapts code to properties of the matrix structure and machine.

Yelick's proudest accomplishment is the recognition of her work on Partitioned Global Address Space Languages (PGAS), which lets users read/write data anywhere in the system and incorporates partitioning for accelerated communication.

"Twenty years from now, I hope programming languages better hide complexity in supercomputers, and that it will be easier to write good programs," Yelick says.

Meanwhile, Yelick is collaborating with her husband and UC Berkeley colleague Jim Demmel on a paper called "Communication-Avoiding Algorithms." Says Yelick, "Avoiding communication is not the right model for marriage, but it is great for parallelism, where communication slows you down."

—*Laura DiDio* is principal at ITIC, a Boston IT consultancy.

Technology | DOI:10.1145/2507771.2507777

Tom Geller

Making the Internet Safe for Gadgets

Initiatives favor direct connections, named resources, and cryptography.

People have predicted the Internet's death by traffic since its origin. Small protocol changes have prevented congestive collapse throughout the years, even as the Internet's fundamental host-to-host structure has remained unchallenged. But that may need to change soon, as an increasing number of sensors, phones, and other mobile devices connecting to the Internet threaten the network's security and reliability.

The growth is staggering: a Visual Networking Index Mobile Forecast by Cisco Systems estimates that mobile traffic worldwide will increase at a rate of 66% per year, or almost three times that of fixed-IP (Internet Protocol) machines, over the period 2012–2017. The nature of the data is also changing, pulled on one side by monolithic files like movies, and on the other by active streams of small data such as those created by field sensors. Security issues compound the problem, as the data travels in a network that protects the endpoints rather than the data packets themselves. Efforts to scale the Internet will need to take such scaling and security effects into account to preserve its usefulness for years to come.

Several active projects are attempting to design a "future Internet" to meet the challenges of the 21st century, including responding to the effects of mobile-device growth. While their implementations vary, they generally share the assumption that yesteryear's host-to-host networking design is no longer appropriate in a world where devices change locations, network connections, and configurations frequently.

One large-scale coordinating effort can be found at the U.S. National Science Foundation (NSF), which announced its Future Internet Architectures (FIA) program in 2010, in addition to announcing funding for three projects that propose ways to augment (and possibly replace) parts of the Internet in ways that are likely to benefit device access directly. (A fourth "NEBULA" project focuses on creating a reliable cloud utility; a fifth, "ChoiceNet," examines economic issues that are relevant to all these projects.)

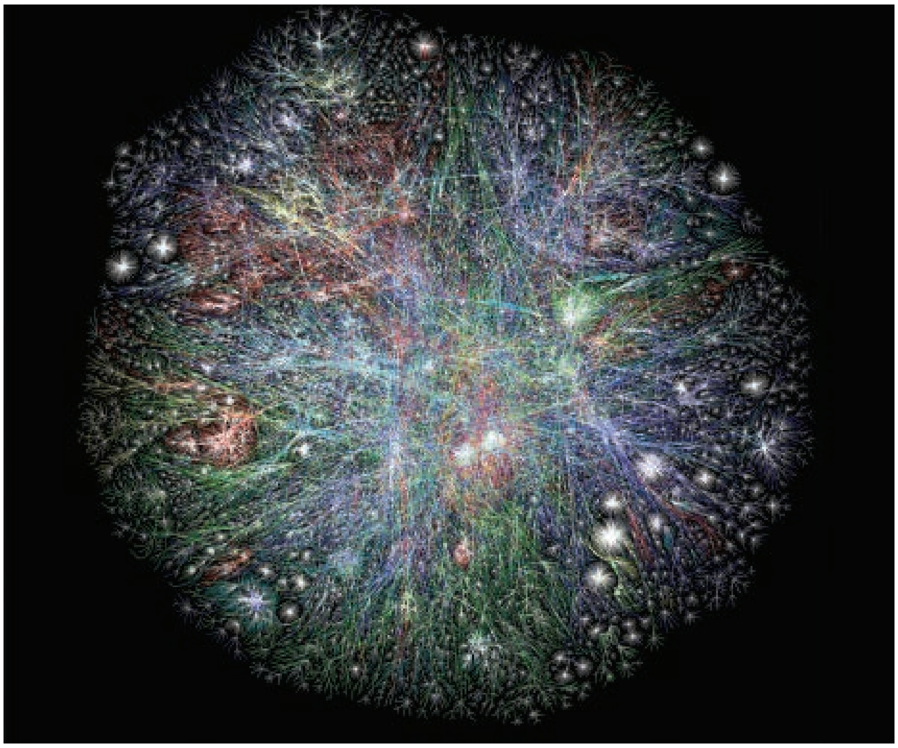


IMAGE COURTESY OF THE OPTE PROJECT

The FIA program has a long pedigree. It grew out of a "Future Internet Design" program that funded more than 50 projects between 2004 and 2009; that, in turn, came from the agency's network architecture work reaching back to NSFNET, the National Science Foundation Network, in the mid-1980s. According to Darleen Fisher, program director for Networking Technology and Systems (NeTS) at the NSF, some of the current problems find their origins in requirements of that early network. "The purpose of NSFNET was to connect researchers and facilities," she said. To speed adoption, "The early NSFNET team decided not to require security, because they were concerned that one more requirement might be the final straw that would keep universities from connecting to it." Such decisions, while appropriate at the time, would come back to haunt Internet users years later.

From Host-Centric to Content-Centric Networking

One family of emerging solutions focuses on the purpose and content of data, rather than on where it lives. The approach is known as information-centric networking (ICN), a topic in which "there are easily two or three dozen projects going on," according to Glenn Edens, research director in the Computer Science Laboratory at PARC.

His group has developed a protocol specification of ICN named Content-Centric Networking (CCN), and with it an open-source reference implementation, CCNx. "That codebase is used by a couple of hundred institutions," said Edens. "It's been ported to around three dozen architectures that we know of. Today, it is running on everything from Raspberry Pis and BeagleBoards and tiny home routers, all the way up to really large cloud switches."

Another project that originally used the CCNx code is the Named Data Networking (NDN) project, an FIA-funded effort led by the University of California, Los Angeles (UCLA) with cooperation from eight other universities and PARC. According to Lixia Zhang, professor of computer science at UCLA and a member of the board of the Asia Future Internet Forum, the NDN project aims to create ICN-style applications "to meet actual needs."

One such need was found in an unlikely place—the University's School of Theater, Film and Television. Said Zhang, "When one of our investigators tried to automate stage control stuff in the early 2000s, he noticed that there was a big gap between what application developers would like and what the Internet actually provides. When he wanted to turn on the left upper-corner lights, the network said, 'give me a packet with an IP address and port number, so I can send it to the destination,' but he did not know or care about IP addresses or port numbers; he wanted things to happen according to the application's semantics."

NDN claims to accomplish this by naming data rather than endpoints, using a human readable and hierarchical name, much like a URL. A user requests data by sending out an "Interest packet;" for example, /tvchannel/videos/favoriteshow/season1episode4.mpg/2ndAct. Routers note the interface of the Interest packet and forward it according to information in a "Forwarding Information Base" (FIB), which is dynamically populated (FIBs are unlike today's routers in that they store name prefixes instead of IP prefixes). When the Interest packet reaches a location that holds the requested data, that location returns the data to the user along the same path. The data packet contains the data's name, its content, and a signature that binds the name to the data.

This architecture removes redundant retrieval of the same content. Since each data packet carries both a name and a signature, it can be cached in various points along the path. The project claims this facilitates content distribution, multicasting, transfer over poor connections, and support for mobility and disruption-tolerant delivery. Zhang claims that, regardless which node supplies the data, NDN gives packets provenance and integrity by binding their names and data together with the signatures.

Billions of Tiny, Moving Targets

In contrast to NDN's hierarchical naming structure, the FIA-funded MobilityFirst project describes a two-part naming system for "network objects," which could be devices, people, internal networks, or even content. First, each network object is given a human-readable name; then, a name certification service (NCS) connects each of these to a Globally Unique ID (GUID). These GUIDs are unstructured and, as such, are unlike either IP addresses or MAC addresses. The result, according to MobilityFirst literature, is a network better suited for the billions of phones, wireless sensors, vehicles, and machines that have no fixed location.

When a network object becomes active, a global name resolution service (GNRS) dynamically binds its GUID to its current network address, or even to multiple network addresses.

Dipankar Raychaudhuri, the project's lead investigator and director of the Wireless Information Network Laboratory (WINLAB) at Rutgers University, claims that several of MobilityFirst's special benefits come from the combination of a new naming procedure and a new routing layer. "We have optimized the routing layer to do new things like network storage functionality," he said, "and it does multi-homing and multi-casting very efficiently. You could also put the [named] part of our architecture on top of ordinary IP for backward compatibility, but then you do not have the benefit of some of MobilityFirst's features such as storage routing, which can be useful for content and cloud applications."

MobilityFirst bindings between

Milestones

Computer Science Honors

CRA HONORS NEUMANN'S DISTINGUISHED SERVICE

Peter G. Neumann, principal scientist at SRI International, has been named recipient of the Computing Research Association (CRA) 2013 Distinguished Service Award, in recognition of contributions to advance computing during a half-century of service and dedication.

According to the CRA, Neumann "has led and driven the fields of computer-related risk and socially responsible use of information technologies. These activities have had an enormous impact on computer science research and remain at the forefront today."

Neumann has been with SRI's Computer Science Laboratory since 1971. He moderates the ACM Risks Forum, and chairs both the ACM Committee on Computers and Public Policy and the National Committee for Voting Integrity. Neumann is a Fellow of the ACM, IEEE, and AAAS, and is also an SRI Fellow.

SURA CITES VARDI FOR 'FOSTERING EXCELLENCE'

Moshe Y. Vardi, Rice University's Karen Ostrum George Distinguished Service Professor in Computational Engineering, recently received the 2013 Distinguished Scientist Award from the Southeastern Universities Research Association (SURA). The award honors a research scientist whose work fulfills the SURA mission of "fostering excellence in scientific research."

Said Charles W. Steger, president of Virginia Tech and chair of the SURA Council of Presidents and Executive Committee, "As one of the top 40 cited computer scientists, Dr. Vardi's contributions to the field have enabled greater science to occur, which distinguishes him for this prestigious award."

Vardi is author or co-author of more than 400 articles on logic and computation, and has co-authored two books. He also is editor-in-chief of *Communications of the ACM*.

names and numbers can be domain-specific. "The name certification service is somewhat similar in spirit to what ICANN provides for domain names," said Raychaudhuri, "but there's no central authority, so you can accept any NCS that you trust. For a device that's part of my car, the NCS could be the auto manufacturer."

Raychaudhuri believes the use of practically inexhaustible GUIDs will prevent the sort of unstable and "roundabout" process now needed to get mobile devices on the Internet. "If you have a smartphone, you get a private IP address from the cell provider, then there is a public IP address in a gateway at the boundary of the cellular network. That gateway is a potential bottleneck or single point of failure. With MobilityFirst, a mobile device does not have to go through the gateway. No matter what network you are connected to, the global name resolution in MobilityFirst will find your current location."

Addressing Unknown Unknowns

MobilityFirst prioritizes issues of mobile accessibility; NDN focuses on content. A third FIA-funded project takes an agnostic approach in forecasting the Internet's needs. The eXpressive Internet Architecture (XIA) project divides Internet players into "principals" such as hosts, content, services, and users. While identifying such current principals, it acknowledges that tomorrow's Internet may have other types of principals, and is built to adapt when they appear. Whatever the principal type of a network resource, XIA describes a way to reach it directly through a new eXpressive Internet Protocol (XIP). As principal investigator and Carnegie Mellon University professor Peter Steenkiste said, "If we look forward to 10 years from now, are we really sure that content and services will be the hot commodity? That is absolutely not clear: there might be a change in the types of entities that people will want to send."

Steenkiste described how an XIA requestor packet could get a satisfying response even when points on the network do not understand its new-form destination type. "The current Internet thinks of packets as having a destination address with one type of ID, such as an IP address," he said. "In XIA, I would request a piece of content with a Content ID. But I would realize that not all Internet routers recognize that, so I would include a second ID, which is a Host ID. If the router does not understand the first one, it can look at the second or third, and it will basically act on the first one that it recognizes. For backward compatibility, that last ID could be an IP address."

"If we continue to solve the problems that we have set out to solve, you should be able to interact with the content network just like you interact with the Web."

For security, XIA borrowed an idea from a prior NSF-funded project, Accountable Internet Protocol (AIP): incorporating a cryptographic ID in addresses so they can be authenticated through a public-private key challenge. Steenkiste says the project uses this scheme so it does not have to rely on key-management infrastructure, as both NDN and MobilityFirst do.

He also explained how such embedded authentication allows duplication not only of content, but also of dynamic services like entire websites. "You could replicate services using a public key, but then you could end up replicating a private key on a lot of devices. In XIA, there is one public-private key pair for the global service; then, there are basic public key pairs with each instance. They are all tied together with a certificate that is issued by the service's owner, not a global certificate authority. The instances cannot hand the service off to others because they cannot sign for it: their private keys are not authoritative."

The Growth of Everything

Three themes are consistent in these projects' approaches: the Internet is growing in terms of not only numbers, but also diversity; the current seven-layer OSI model needs adjustment to move it away from host-centric networking; security must be a first-order concern in every data transaction, not only at the endpoints. These changes require fundamentally different ways of thinking about networking from those currently in play.

Yet for Internet users, the basic experience should remain the same. As PARC's Edens put it when speaking of CCN, "The seven-layer model is getting pretty darn tired, but if we continue to solve the problems that we have set out to solve, you should be able to interact with the content network just like you interact with the Web. You ask for content, and you get it back — you do not care whether it is cached or comes from the original publisher. But with intelligent routers, you can have things like dynamic rerouting, error recovery, and broadcasting. You can put all these functions into the network as services."

Or, as NSF's Fisher noted, "We never expected the FIA projects would replace the current Internet, although some project evangelists are motivated by that as a holy grail. In either case, these projects can have a very large impact. That is very exciting." C

Further Reading

NSF Future Internet Architecture Project, National Science Foundation, http://www.nets-fia.net

VNI (Visual Networking Index) Mobile Forecast Highlights, 2012-2017, Cisco Systems, Inc., http://www.cisco.com/web/solutions/sp/vni/vni_mobile_forecast_highlights/index.html.

Raychaudhuri, D., Nagaraja, K., Venkataramani, A.
MobilityFirst: A Robust and Trustworthy Mobility-Centric Architecture for the Future Internet. Draft, August 2012. http://mobilityfirst.winlab.rutgers.edu/documents/MobilityFirst_paper_MC2R_v10.pdf

Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J. D., Smetters, D. K., et al. (2010)
Named Data Networking (NDN) Project. PARC Tech Report 2010-003 NDN-0001, PARC.

Han, D., Anand, A., Dogar, F., Li, B., Lim, H., Machado, M., et al. (2012)
XIA: Efficient support for evolvable internetworking. Proc. 9th USENIX NSDI.

Tom Geller is an Oberlin, Ohio-based science, technology, and business writer.

Society | DOI:10.1145/2507771.2507778

Paul Hyman

Software Helps Linguists Reconstruct, Decipher Ancient Languages

Linguists who once spent an entire career reconstructing a major language family now can accomplish that in just a few hours.

LINGUISTS WHO RECONSTRUCT ancient languages—and who previously did the arduous work solely by hand—now have another tool in their arsenal to speed up their laborious efforts. Computer scientists have proven they can use software to recreate the early languages from which modern tongues have derived.

While previously it might have taken a linguist their entire career to reconstruct a major language family, now software running computations on, say, a large experiment that may involve a sixth of the world's languages can be completed in just a few hours.

The achievement is not about speed, cautions Dan Klein, associate professor of computer science at the University of California, Berkeley. It's about being able to do things in a large-scale, data-driven manner without losing all the important insights that historical linguists have gained in working on these sorts of problems for decades, he says.

Indeed, linguistic researchers compare these techniques to those used for gene sequence evolution.

"This achievement should not be compared to, for example, Deep Blue, IBM's chess-playing computer," Klein insists. "This is not a human-versus-machine story in which humans used to be better until finally a computer was able to take the crown. This is a story of computation giving human linguists new tools that supplement their weaknesses and let them work in new ways."

The efforts of Klein and his colleagues are described in their paper, "Automated Reconstruction of Ancient Languages Using Probabilistic Models of Sound Change," published by the National Academy of Sciences in February.

PHOTOGRAPH BY REMY MUSSER

According to Klein, the work's main contribution was a new tool that researchers can use to automatically reconstruct the vocabularies of ancient languages using only their modern-language descendants.

The goal, he says, is not to just rewind the clock; rather, it is to better understand the processes that give rise to language change, and to model how the evolution of language proceeds. "And so we want to know things like what kind of sound changes are more likely and what kind of sound changes go together," he explains.

To test the system, the team applied it to 637 languages currently spoken in Asia and the Pacific, and recreated the early language from which they all descended. The result? More than 85% of the system's reconstructions were within one character of the manual reconstruction provided by a linguist who specialized in Austronesian languages—and, of course, the differences are not necessarily errors.

How does the system work?

The way we produce words differs from the way our ancestors pronounced those same words. As time goes by, minute, ongoing alterations help turn an ancestral language like Latin into modern descendants like French, Italian, and Portuguese.

The sound changes are almost always regular, with similar words changing in similar ways, explains Klein, and so patterns are left. The trick is to identify those patterns of change and then to "reverse them," basically tracking the evolution of the language backward in time.

"Linguists have known this for a good 100 years or more, but it's a hard and time-consuming process to do by hand," he says. "However, that is where computers shine."

Yet the use of computers and linguistic software is not limited to *reconstructing* ancient languages.

Ben Snyder, an assistant professor in the Department of Computer Sciences at the University of Wisconsin–Madison, employs his own software to do *decipherment* of some sort of text—perhaps a tablet—written in a long-dead language that may or may not be related to a living language. He then tries to reconstruct the dead language, making a prediction about that language for which he has no direct evidence.

In his most recent work, he developed a software program into which he is able to feed an unknown language not necessarily connected to any other language. The program is then able—in about 30 seconds—to "say something useful" about the language, says Snyder.

For instance, in a paper for this year's Association of Computational Linguistics (ACL) Conference in Bulgaria entitled "Unsupervised Consonant-Vowel Prediction Over Hundreds Of Languages," Snyder describes how his program is able to tell—with 99% accuracy—which letters of a long-dead language are consonants and which are vowels. In addition, the paper describes how the program can determine—with 89% accuracy—some of the qualities of the consonants; for example, which are probably nasal sounds and which are not.

"What I wanted to develop was a program based on machine-learning techniques that would examine hundreds of languages and, by doing so, build a universal model of linguistic plausibility," he explains. "What I've done so far is just the starting point; I hypothesize that, in time, we'll be able to determine much more with the software."

"What I wanted to develop was a program ... that would examine hundreds of languages and, by doing so, build a universal model of linguistic plausibility."

Unlike the reconstruction program created by Dan Klein's team, which acts to supplement the manual work of linguistics by making it simpler and more efficient, Snyder says what his decipherment software "goes way beyond what humans are able to do through manual analysis."

While Snyder does not see software replacing more-traditional manual linguistic methods completely, he suspects the field will undergo a shift toward greater use of computational methods, given the amount of data that is being accumulated.

"This is a really nice example of analyzing large amounts of data using novel algorithmic techniques," he says, "and a great example of computer science being the new handmaiden of the sciences, the way math once was."

Why would any linguistic analysis continue to be done by hand, when computational techniques seem to be so much faster and easier?

Kevin Knight believes that humans are simply better at finding new data

EU Policy

ACM Europe Backs Software as Enabling Tech

Newly incorporated ACM Europe recently aired its support for a report that identifies software as a Key Enabling Technology (KET), and urged its implementation as part of the Horizon 2020 strategy to secure Europe's global competitiveness.

The report, "Software Technologies: The Missing Key Enabling Technology," was issued by the European Commission's Information Society Technology Advisory Group (ISTAG) in 2012 in response to two EC studies that purported to identify KETs "that strengthen the EU's industrial and innovation capacity to address the societal challenges ahead and proposes a set of measures to improve the related framework conditions."

The 2009 EC communication "Preparing for Our Future: Developing a common strategy for key enabling technologies in the EU," and the EC's June 2011 Final Report of the High-Level Expert Group on Key Enabling Technologies, both identify six enabling technologies that "underpin innovation in many strategic sectors and play a key role in making new products and services affordable for the population at large:"

- **Nanotechnology**
- **Micro and nanotechnology**
- **Industrial biotechnology**
- **Photonics**
- **Advanced materials**
- **Advanced manufacturing systems**

The 2012 ISTAG report says software "is a key driver for the European economy," and suggests a "Strategic Agenda for Software Technologies in Europe should be created in cooperation with industry, academia and public sector."

ACM Europe chair Fabrizio Gagliardi says, "All six KETs identified in the previous studies rely on software. The creation of the KETs, and the engineering, production, and distribution of new products based on them, all require software."

A letter to EC chief scientific officer Anne Glover by Gagliardi, ACM Europe vice-chair and former ACM president Wendy Hall, and ACM Europe Council member and ACM vice president Alexander Wolf, notes, "It would be a serious missed opportunity if the considerable funding that Horizon 2020 will provide to European researchers ended up most directed at supporting only applied computing science... we believe that support for fundamental research and education in computing science should become a priority in Horizon 2020. This is essential to support the innovation life cycle in computing, which will eventually allow major society challenges of Europe to be addressed in the years to come."

With an €80-billion ($106-billion) budget (of which €24.5 billion/$32.5 billion is dedicated to strengthening the EU's position in science), the Horizon 2020 program will run 2014–2020 as part of a drive to create growth and jobs in Europe.

—Lawrence Fisher

and seeing the patterns in that data. Knight is a senior research scientist at USC/Information Sciences Institute.

"On the one hand, computers are much more thorough and much more patient than people are at searching for patterns," he says, "but they only look for what you tell them to look for. If the text uses some other method of encoding that you didn't tell the computer about, it's not going to find an answer. Humans, on the other hand, are much better at this kind of flexible pattern-matching and adapting."

For example, he says, there are multitudes of ways to write the letter 'A' in English, including capital, lower-case, cursive, and so on.

"I could show you 50 different ways and you would look at them and say, 'yeah, that's right, they are all A's'— different from each other but recognizable as A's," he says. "But while humans can do that naturally, it's difficult to program computers to do it—although they are getting much better at it."

He predicts the decision whether to do linguistic work by hand or software will depend on the specific issue under consideration, "although in much of our work, a joint human-computer team tends to be the best way to go."

For instance, three years ago, Knight teamed up with Ben Snyder and Regina Barzilay, an associate professor in MIT's Computer Science and Artificial Intelligence Lab, to present the paper "A Statistical Model for Lost Language Decipherment" at ACL 2010. The paper demonstrated how some of the logic and intuition of human linguists can be successfully modeled, allowing computational tools to be used in the decipherment process.

"Through a painstaking trial-and-error process, scholars were able to decipher Ugaritic, a 3,000- to 5,000-year-old language from ancient Syria known almost only in the form of writings from ruins," says Knight. "It took them five years to do that by hand."

In their 2010 paper, Knight and his co-researchers described how it took them six months to develop a computational model to do the same task—and about an hour to run it and achieve results.

The team then evaluated their software by comparing those results to what the linguists had achieved by hand.

"Computers have totally taken over in that area of biological classification and, I predict, they'll totally take over in the area of linguistic reconstruction, for sure."

They got 29 out of 30 letters correct, missing only one "rare letter." They recovered 60% of the Ugaritic words that have a similar meaning in Hebrew and come from a common ancestor so they have a similar pronunciation.

Snyder believes the project illustrates how the field of linguistics will undergo a shift toward greater use of computational methods, but those methods "will be guided by our knowledge of linguistics and what are the relevant features of language to look at. So, in a sense, the design of the algorithm still needs to be guided by human linguistic knowledge."

Knight concurs that the future of linguistics will definitely depend on software—and how much of the data regarding languages can be assembled online and available to computers.

"The big enabler will be getting all this data online, organizing the databases, and allowing computers to analyze it all," he says. "There's a clear parallel here to DNA sequencing and biological data analysis. Computers have totally taken over in that area of biological classification and, I predict, they'll totally take over in the area of linguistic reconstruction, for sure."

As for Dan Klein and his NSF-funded reconstruction efforts, he is preparing for the next steps, which include further scaling up the current models so that he and his team can reconstruct even further back than the 7,000 years they've been able to so far. That's a matter of gathering more data—larger collections of languages that are even more distantly related—and, at the same time, tweaking the software, since the further back you want to go, the better the models have to be.

For example, he would like to feed in all the languages of the world and then draw inferences about what their roots looked like.

"Obviously, because so much data is involved, it will require computation, not just manual work," says Klein. "But a historical linguist would observe that what we are attempting is not to automate what people have been doing by hand, because people are very good at the kind of research they do by hand. They would say that tools like ours give us a way to answer new kinds of questions that are impractical to answer by hand." C

Further Reading

Bouchard-Cote, A., Hall, D., Griffiths, T., and Klein, D.
"Automated Reconstruction of Ancient Languages Using Probabilistic Models of Sound Change," March 12, 2013, the National Academy of Sciences of the United States of America, http://www.pnas.org/content/110/11/4224

Kim, Y., Snyder, B.
"Unsupervised Consonant-Vowel Prediction Over Hundreds Of Languages," to be published at the summer 2013 Association of Computational Linguistics Conference, http://pages.cs.wisc.edu/~bsnyder/papers/consvowel-acl2013.pdf

Snyder, B., Barzilay, R., and Knight, K.
"A Statistical Model for Lost Language Decipherment," July 13, 2010, the 2010 Association of Computational Linguistics Conference, http://people.csail.mit.edu/bsnyder/papers/bsnyder_acl2010.pdf

Bouchard-Cote, A., Griffiths, T., Klein, D.
"Improved Reconstruction of Protolanguage Word Forms," May 31, 2009, the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, http://www.aclweb.org/anthology-new/N/N09/N09-1008.pdf

Hall, D. and Klein, D.
"Large-Scale Cognate Recovery," July 27, 2011, the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11), http://www.aclweb.org/anthology-new/D/D11/D11-1032.pdf

"Kevin Knight: Language Translation and Code-Building" (video), April 18, 2013, https://www.youtube.com/watch?v=bcfOT-jFazc

Paul Hyman is a science and technology writer based in Great Neck, NY.

Recap | DOI:10.1145/2507771.2507785 Sarah Underwood

The Alan Turing Year Leaves a Rich Legacy

A year-long celebration of the life and work of a man whom many call the founding father of computer science.

THE 2012 ALAN Turing Year celebrating the life and work of the man many call the founding father of computer science was a triumph. Around the world, it brought together academics developing Turing's uncompleted work, inspired school students with early visions of computing, and reached out to the wider public with a vast array of events that brought the character and achievements of the quiet genius to life.

The year highlighted not only Turing's practical success, particularly his part in cracking German Enigma codes at Bletchley Park during the Second World War, but also brought into focus some of the concepts he conceived across disciplines, including mathematics, computing, computer science, informatics, morphogenesis, and philosophy.

Barry Cooper, a professor in the School of Mathematics at the University of Leeds and chair of the Turing Centenary Advisory Committee, started to encourage colleagues to contribute to the Turing Year five years before it began on January 1, 2012. He went on to coordinate the year's activities.

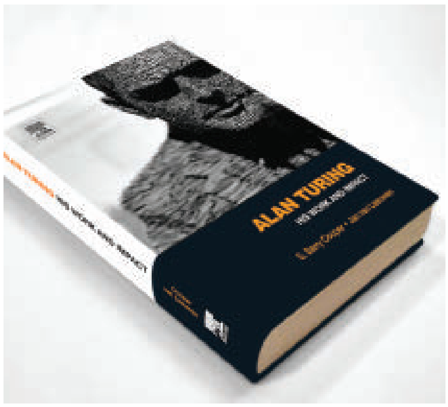
One of these was the Turing Centenary Conference held at Cambridge University in June 2012 and entitled "Computability in Europe (CiE) 2012 — How the World Computes." Cooper explains, "The mission of this event was to address concerns about how science was fragmenting. We wanted to return to more joined-up thinking about computability and how it affects everyone's life.

"The Turing Year was important in highlighting the need for fundamental thinking."

"More generally, too, the Turing Year was important in highlighting the need for fundamental thinking. Looking back, the year was amazing. It included more events than we had expected, it brought forward consideration of extended forms of computation, and it opened a window on things that many people find difficult to understand."

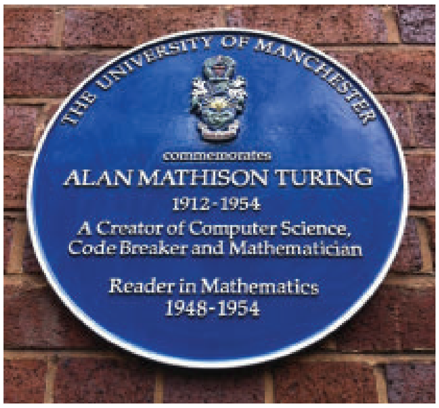
Leslie Valiant, professor of Computer Science and Applied Mathematics at Harvard University, and the 2010 ACM A.M. Turing Award recipient, spoke at several Turing Year events. He suggests the centenary events were a culmination of a growing realization among scientists of the importance of Turing's work, rather than a rediscovery of his brilliance. He explains, "Turing may be a newly recognized celebrity, but this is not necessarily the result of the centenary celebrations, as his influence on the world has become self-evident over the past decade in a way that it was not evident 20 to 30 years earlier. The Turing Year may have enhanced recognition of Turing's work by bringing together many scientists who did not know individually just how many others also traced their work to Turing's ideas. It also helped many realize just how sweeping Turing's influence has become."

The breadth of Turing's thinking and the observations he did not go on to develop, but which now have great meaning in the world, are part of what made Turing a unique and remarkable man. At the same time, however, the complexity of his thinking and the difficulty in explaining to the layman the concepts he proposed, have



The cover of the recently released book, *Alan Turing: His Work and Impact*.

An Enigma cake at the Turing Centenary Dorkboat cruise. Photo: Normal Flora.


THE UNIVERSITY OF MANCHESTER
commemorates
ALAN MATHISON TURING
1912-1954
A Creator of Computer Science,
Code Breaker and Mathematician
Reader in Mathematics
1948-1954
A plaque at Manchester University, where Turing worked. Photo: Peter Hughes.

At a commemorative event, lasers flash in Morse code "Thank you" to Alan Turing and all who served in the Battle of the Atlantic. Art: Craig Morrison and Joel Cockrill. Photo: Stephen King.

helped deny Turing the public recognition given to other great scientists such as Charles Darwin, Isaac Newton, and Albert Einstein.

This is disappointing to scientists following in his footsteps, although they do acknowledge that the Turing Year was successful in raising Turing's profile to a limited extent. Christos Papadimitriou, C. Lester Hogan Professor of Computer Science at the University of California, Berkeley, says, "One outcome of the centenary celebrations is that Turing is a little more in the public mind; I see fewer blank faces when I mention his name. This is a good thing, as Turing is my hero. His opus is nothing short of brilliant and he has made more impact than anybody else. He has changed the world in ways that gravity did not, and he matches Darwin in terms of razor-sharp, original thinking in the face of an intellectual community with opposing thoughts. Darwin, Newton and Einstein are giants, but so is Turing, and I hope he will gain a profile like them."

In discussing Turing's accomplishments and the celebration of his birth in London on June 23, 1912, it is impossible to disregard the U.K. government's proposal to pardon Turing posthumously for his criminal conviction for homosexuality in 1952, which is believed to have contributed to his suicide two years later. For many of those involved in the Turing Year, this and subsequent media coverage have been a distraction and discordant note in the proceedings, although they have, ironically, served well in raising the profile of the man in the scientific spotlight. While many agree that Turing was hurt by the humiliation of the conviction and would, in his quiet way, have appreciated a pardon, the main concern is that a pardon and its media coverage should not be the lasting legacy of Turing and the Turing Year.

With legislation expected to be in place to support a pardon either at the end of this year or early next year, and the 2012 Turing Year spilling into 2013 and beyond, a larger legacy is described by Cooper. "The future is not computable, so we can't see what will happen, but the Turing Year has increased interest in basic issues of computability and engaged people who didn't previously know much about Turing. One spin-off is that as Turing becomes better known, people will look at the context of his work and begin to discover others with an early interest in computing, such as John von Neumann."

Cooper also suggests the Turing Year could encourage more computer scientists and mathematicians to write for a wider public audience, while Papadimitriou hopes the year will spark greater interest in computer science education and research.

As well as such aspirations, the legacy of the Turing Year includes a number of artifacts, some of which, such as the DVD of *Codebreaker* (an award-winning drama-documentary film on Turing's life), will serve to keep him in the public eye, while others, such as Cooper and Jan Van Leeuwen's book entitled *Alan Turing: His Work and Impact*, will help sustain scientific interest in his achievements for many years to come.

Sarah Underwood is a technology writer based in Teddington, U.K.

Turing Centenary Advisory Committee chair Barry Cooper at a Sao Paolo exhibition.

A performance of "Turing: A Staged Case History" in Milan. Photo: Diego Ronzio.

Hong Kong Turing fans celebrate his 100th birthday on June 23, 2012. Photo: Cpak Ming.

DOI:10.1145/2505337

Michael A. Cusumano

Technology Strategy and Management
Evaluating a Startup Venture

Considering the key elements of successful startups.

STARTUPS ARE AN engine of economic renewal and change around the world (see "Dealing with the Venture Capital Crisis," *Communications*, Oct. 2009). But successful startups are rare, and startups that go public and yield strong financials like Facebook are even more extraordinary (see "Reflecting on the Facebook IPO," *Communications*, October 2012). For example, living MIT alumni created 26,000 active firms with 3.3 million employees and annual revenues of nearly $2 trillion as of 2006. Five to seven years after their founding, however, only 30% of MIT startups were successful (approximately 60,000 failed).[5,6] The National Venture Capital Association says about 75% of startups succeed, but a recent Harvard Business School study found this true of only about 25%. Stricter definitions of return on capital suggest only 5% of startups succeed and merely 1% go public.[2]

It should be possible for potential investors as well as would-be entrepreneurs to evaluate startup ventures more systematically. This column attempts to help them do this with a short checklist of key elements to look for. It is based on many years of working with startups and a list earlier published in *The Business of Software* (2004), with some additional reflections and examples.

1. A Strong Management Team

Venture capitalists often say *they invest primarily in people—the entrepreneur or the management team—and secondarily in ideas*. Some reverse this order, investing in ideas first and people second. Ideas, such as beliefs about sectors or technologies that will be important in the future (for example, social media, location-specific applications, healthcare software) are worth little without a team to execute the plan successfully. People end up being key in any case. A strong management team has the right level and breadth of experience, and needs strong technical leadership if it is a technology-driven company. At the same time, ventures dominated by technology often spend too much money refining the product and too little effort getting ready for customers and closing deals. Especially with technology startups, success often depends on having founders with solid marketing or sales expertise.[5,6]

It should be possible for potential investors as well as would-be entrepreneurs to evaluate startup ventures more systematically.

2. An Attractive Market

Successful startups usually focus on markets capable of becoming large, fast growing, and profitable for new entrants. Whether "horizontal" (for example, everyone with a computer or a smartphone is a potential customer) or "vertical" (for example, every financial services company is a potential

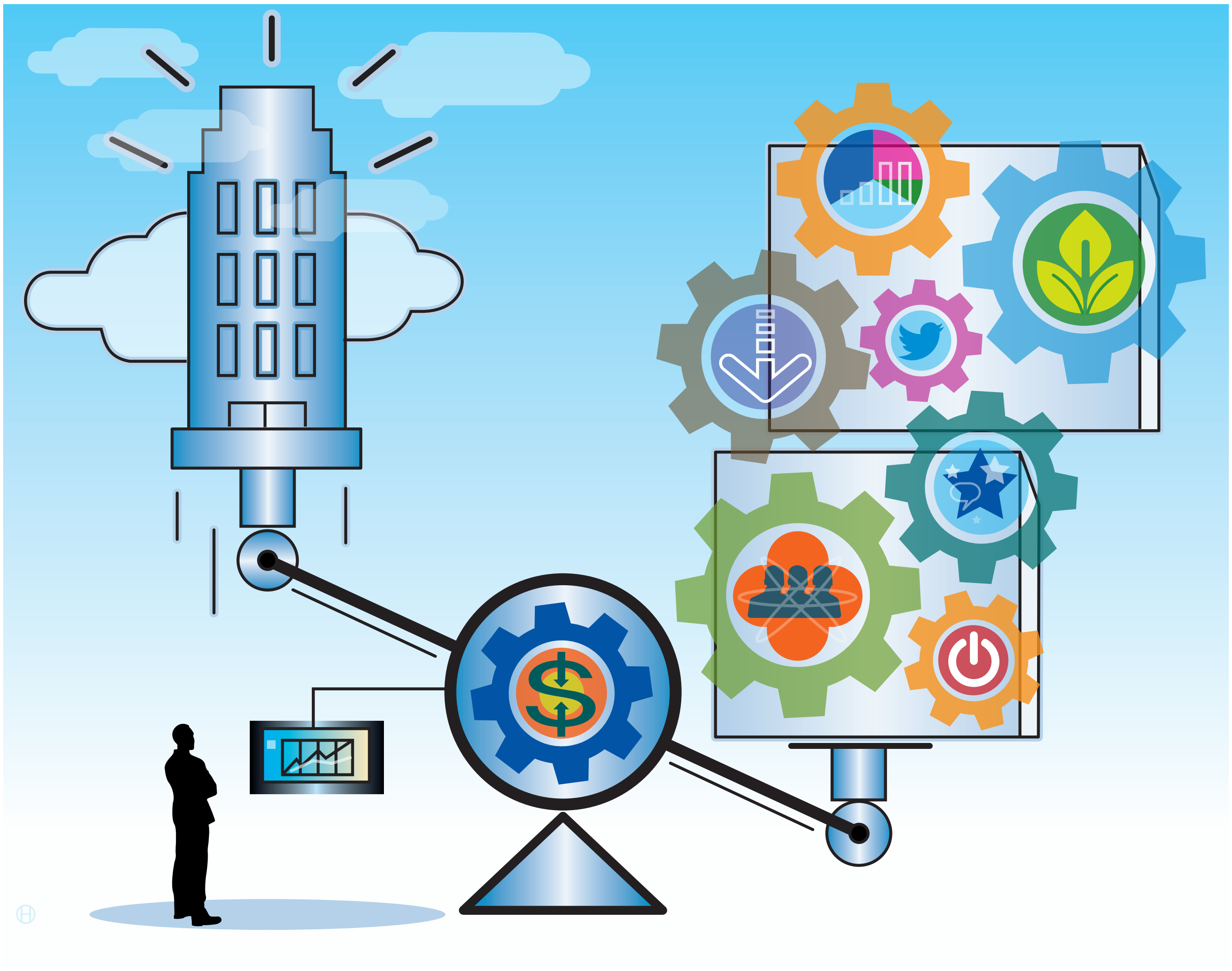


ILLUSTRATION BY HANK OSUNA

customer), they must be *structurally attractive*. This means there should be high enough entry barriers to keep out new competitors once you enter. Rivalry should not devolve into cutthroat price wars because of too many competitors or one firm that gives away the product for free. Neither buyer power nor supplier power should be strong enough to negotiate prices downward too easily. There should not be good substitutes for the basic product or service.[4] These "five forces" were made famous by Harvard professor Michael Porter. Many startups also require "complementary" products (such as software applications for a new hardware platform) or infrastructure elements (such as Wi-Fi availability)—what Andy Grove of Intel called the "sixth" industry force.[3] If needed, these additional factors must also be available for a startup to succeed.

Horizontally packaged Software as a Service (SaaS) offerings can be relatively easy to scale up, but they usually have high customer churn, high costs of customer acquisition, and small monthly payments from customers who can easily cancel. As a result, these businesses can take many years and lots of funding to become profitable. Even Salesforce.com, a SaaS pioneer, needed vast amounts of venture capital and many years of effort before it created a large installed base, and it still has difficulty earning a profit. Horizontal markets also attract a lot of competition because they are so large. A startup can spend all its money trying to be everything to every customer in such markets (see "Beware the Lure of the Horizontal," *Communications*, July 2003). Professional service firms targeted at specialized vertical segments (for example, tailoring enterprise software products for financial services or the retail industry) are easier to establish and run at a profit, but they can be difficult to scale because revenues and headcount usually grow in a one-to-one ratio.

The wrong way to think about a market opportunity is to describe a huge industry (like U.S. healthcare or business intelligence software) and argue that the startup needs to get only, say, half of 1% in order to be viable. Size alone does not make a market attractive. Structural factors are more important. Then the entrepreneurs need to convince investors of their particular advantages over competitors, and how *their* firms will reach customers and capitalize on those advantages. High failure rates suggest *the most likely percentage a startup will get of any market is zero!*

3. A Compelling New Product or Service

A *compelling* product or service appears as a "must-have" to a specific type of customer. Some entrepreneurs have a deep familiarity with a market and are able to identify such customer needs that are unfilled or poorly met. At earlier times, Microsoft did this with PC programming languages, Netscape with a mass-market browser, and Apple with the Macintosh, iPod, and iPhone. Some entrepreneurs create companies around a product or service they themselves desperately want, as Steve Wozniak did with the first Apple computer and Steve Jobs did with the iPad. But attracting funding from professional investors usually requires more than emotion: They want quantitative and qualitative data demonstrating the superior benefits of the new product or service as well as what potential users are willing to pay to get it. This *value* to customers ultimately depends on what competing or substitute products and services are available and at what price.

Some startups enter a market with a product or service that is cheaper but less functional than what large, established firms offer. The new offering may seem to offer little value and the venture little potential, but this evaluation may be misleading. Clayton Christensen's 1998 book, *The Innovator's Dilemma*, discusses this specific type of opportunity and threat—when large firms focus too much on existing customers and fail to recognize the threat of new technologies, services, or business models that are initially inferior but improving quickly.[1] Such examples have occurred with small disk drives, personal computers, digital photography, and Internet-based or SaaS applications versus traditional packaged software.

4. Strong Evidence of Customer Interest

Startups need to convince investors that *actual customers are willing to buy the new product or service*. Most entrepreneurs underestimate how difficult it will be to sell beyond "friends and family." Some new companies boast they have lined up many beta users and marketing partners, but these are not as convincing as actual letters of intent to purchase.

A prototype or early product version helps land new customers by allowing them to visualize how the product will work. Product firms that try to get customers or funding without a prototype generally have trouble.[5,6] Service startups do not have physical prototypes, but they can try to convince customers to begin with a limited engagement and then use a small success to indicate they can handle a bigger project.

5. Overcoming the "Credibility Gap"

The "credibility gap" is the *fear among customers that the venture will fail*, leaving the buyer without technical support or a future stream of product upgrades. Since more than 90% of ventures do fail, this fear is real. It is easy for customers to go with an established vendor, even with an inferior or more expensive product. This situation leads to a "Catch-22": a startup must line up paying customers to serve as references for new customers, but new customers will usually not sign on unless the startup has sufficient money to last a significant time period. *The credibility gap may be the most common cause of failure for startups.* To tackle this dilemma, startups can do several things. They can offer large discounts to get those first reference customers, or partner with established firms for long-term support. They can line up investors, advisors, and board members to show the startup is a viable enterprise. Or the startup can package its product or deliver the service in such a way that the customer experiences immediate benefits and does not have to worry about the venture's longevity.

6. Demonstrating Early Growth and Profit Potential

Many investors want to know *how the startup will grow the business and generate enough cash to reach breakeven and maybe even profitability*. These

Size alone does not make a market attractive. Structural factors are more important.

milestones must happen in a timeframe consistent with the available funding. It is impossible to predict whether or when a new firm will actually make any money. But investors need to be extremely wary of business plans that call for many tens of millions of dollars and years of R&D and marketing before the venture is expected to generate any revenue. When too much time and money are required, too many bad things can happen: New competitors can enter and established firms can counterattack. Technologies can become outdated or government regulations can change. Moreover, a venture that has to keep raising new rounds of funding often leaves both the early investors and the entrepreneurs with little equity. Once the startup becomes financially desperate, later investors can impose draconian funding terms. Promising MIT startups such as E-Ink in electronic displays, and A123 Systems in batteries, turned problematic in this way, as did Akamai, which took a long time to build what is now a great business in digital content delivery.

7. Flexibility in Strategy and Technology

Investors look for *focus* in a startup because most have limited resources and time compared to large, established firms. However, most startups also need to demonstrate *flexibility*—in strategy, business models, and technology. Startups often get the product strategy and the business model *wrong* the first time around. Even with multiple chances, they often get the second and third times wrong as well. So both focus and flexibility, which seem contradictory, are often critical to success. The right formula is likely to emerge only over time, through trial and error, rather than through deliberate planning in advance.[a] Startups need to focus their resources on a particular approach, but then be prepared to change course or "pivot" quickly if the initial strategy is not working. To gauge strategic flexibility, investors should talk to the founders and the management team about

Of course, success and failure are easy to explain in retrospect but difficult to predict in advance.

different options. Have they thought about what else they could do with the technology and the skills they are cultivating? For a software startup, a potential investor might also ask how tightly the code is tied to a particular hardware or software platform, and how general-purpose the functionality of the code is.

8. Potential for a Large Investor Payoff

A startup wanting more than "angel investors" or "bootstrapping" with the help of friends and family should offer good prospects to professional investors for a significant payoff within a time frame that is typically no more than seven years. Venture capitalists often look for >20% annual returns on their portfolios, so they are looking for big winners. Not surprisingly, they tend to give most of their money to the better startups, which often do not need money to survive but need investment to get big fast.

It is impossible to know in advance which firms will succeed, but the eight points discussed in this column form a framework to inform this murky question of potential investor payoff. For many venture capital firms, the market opportunity has to be large enough for the startup to become worth at least $100 million. The business model must demonstrate how to scale up sales while maintaining some advantage over the competition. Scaling can be relatively easy (for example, for a packaged software product company, although competition is fierce), relatively difficult (for example, a SaaS company that needs many customers paying those small monthly fees), or extremely hard (for example, a professional services company that is costly and time-consuming to grow because it must hire and train so many people). Automated horizontal services delivered via the Internet, (for example, Google, LinkedIn, Facebook, Pandora Radio, Spotify, FourSquare) can scale quickly because of platform dynamics and network effects (see "The Evolution of Platform Thinking," *Communications*, Jan. 2010). Still, there are no guarantees these firms will build a profitable business or go public. Nonetheless, a large payoff might still come from another "liquidity event" in addition to an IPO such as selling out to a larger firm. We can estimate the potential value of such deals by using the recent sale prices of comparable startups or the market values for comparable firms that are publicly traded.

Conclusion

Of course, success and failure are easy to explain in retrospect but difficult to predict in advance. Potential investors must therefore ask, with each of the eight points mentioned here: *What can I know and how early can I know it?* Would-be entrepreneurs also should think carefully about the items on this list. Unforeseeable factors such as chance events and timing affect all firms. But those startups that can objectively evaluate their potential and improve their weaknesses should be able to increase the possibility of their success. **C**

a See, for example, H. Mintzberg, Crafting strategy. *Harvard Business Review*, July–August 1987, 66–75.

References

1. Christensen, C. *The Innovator's Dilemma.* Harvard Business School Press, Boston, 1997.
2. Gage, D. The venture capital secret: 3 out of 4 start-ups fail. *The Wall Street Journal* (Sept. 19, 2012).
3. Grove, A.S. *Only the Paranoid Survive.* Currency/Doubleday, New York, 1996, 30.
4. Porter, M.A. *Competitive Strategy.* Free Press, New York, 1980, 3–33.
5. Roberts, E.B. *Entrepreneurs in High Technology: Lessons from MIT and Beyond.* Oxford University Press, New York, 1991.
6. Roberts, E.B. and Eesley, C.E. Entrepreneurial impact: The role of MIT—An updated report. *Foundations and Trends in Entrepreneurship 7*, 1–2 (2011), 1–149.

Michael A. Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and School of Engineering and author of *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Unpredictable World* (Oxford University Press, 2010).

The eight factors framework and discussion is adapted from M.A. Cusumano, *The Business of Software.* Free Press, New York, 2004, 195–214.

DOI:10.1145/2505338

Phillip G. Armour

The Business of Software
When Faster Is Slower

How the speed of modern tools may decelerate development.

SOFTWARE DEVELOPMENT IS always about acquiring knowledge. How we get that knowledge can vary, but it is never enhanced by being lazy in thought or deed and there are capabilities in modern systems development that can encourage laziness. It was not so years ago...

When Small Was Big

When I started programming professionally in the early 1970s, the only available computer was a large mainframe. When I say large, I mean physically; in memory size it was miniscule.[a] The computer's primary function was running a steelworks and any software development task requiring computer resources had to wait until the machine was not engaged in more important tasks. This would generally be around 2:00 A.M.

Overnight Delivery

Any computer operation: compile, assemble, or test, could only take place overnight and we had to wait until the next morning to find out if it worked. So every software development activity had to be very, very carefully thought out. Machine runs were terribly time-consuming and extremely expensive—much more than the cost of programmers. So to get the best use out of the scarce machine time we did a lot of desk-checking. We reviewed and inspected everything: documentation, code, tests. We even developed an inspection technique wherein programmers would "play computer" by manually walking through assembly language programs with people taking the roles of registers, accumulators, and memory locations. Test data was written on slips of paper that were passed across the table from memory to accumulators or registers and back again. The program stack was just that: a stack of these pieces of paper. This process was very tedious, it was very slow, and it required a lot of time and a lot of effort. But it made you think.

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

Quick Return

Fast-forward 40 years and things are different. Most of us carry, in our pockets or in our purses, more computing power than existed in the whole North of England in 1972. Chances are you could not find even

a It was an ICL 1901 series with 8k (words) of memory.

one chip as small as 32k in any laptop or desktop you would buy today. There are libraries full of books on languages, methodologies, and development lifecycle models. Modern systems development environments are exactly that: environments. Integrated development environments (IDEs) contain programming language syntax checkers and compilers, editors, functional code and code snippet libraries, object browsers, code navigation and coverage aids, environment support and virtual machine interfaces, authoring tools, configuration management and build functions, even embedded project management capabilities. Critically, IDEs usually contain interactive testing and debugging environments and this can be where problems may arise.

These powerful tools allow us to create *something* that "works" and create it very quickly. From a standing start, IDEs allow developers to build and test complex systems in days if not hours.

This is very seductive.

Modern systems development environments are just that: environments.

Low Energy State

Humans, like most dynamic systems, often attempt to operate at the lowest energy state possible. When the perceived job of a software engineer is to "build something," and the environment provides a very quick way to build something, it is difficult not to follow that road to its end. Fast compilers and interactive debug environments can encourage shortcut approaches that compromise the learning essential to effective systems development. We may build something when we really need to build *the* thing.

There is instant gratification in coding and immediately seeing the code run. It can encourage programmers to throw something together and then play with it to see how (or even if) it works. At its best this can be considered learning by experimentation; at its worst it can be mindless hacking.

Thinking and Experimenting

Don't experiment when you should think; don't think when you should experiment.
—Jon Bentley, computer scientist

There are two ways to approach understanding something. We can think carefully about the required logic to solve a problem before trying to produce something that will show how our understanding holds up. Or we can experiment by running a program in different states of development against different inputs and see what happens. Both are legitimate approaches when used carefully and thoughtfully. But we have entered an era where computing power and speed are so readily available and so cheap that the experimenting approach may be causing us problems; it may encourage programmers to be lazy and that is never a good thing.

When modern IDEs allow us to build something that works very quickly and modern debuggers allow us to execute it before our eyes in real time there is a strong temptation to design and program by trial and error. Programs can be patched together as a sequence of ad hoc amendments to an initially weak and ill-considered design. Once the program seems to work, it may be bundled with other programs similarly designed. Systems built this way do not work very well, are difficult to understand, and are hard to maintain.

Experimenting and Thinking are Different. *Experimenting* tends to be situational and instance-driven. It is highly influenced by our initial (and necessarily imperfect) understanding or simply by how the program was first thrown together. Experimenting also encourages understanding-in-the-small since the focus is usually to more and more detail to try to somehow make *this* program work.

Thinking tends to be more contextual. Using it, we build mental models that allow understanding of the in-

Calendar of Events

October 18–20
11th ACM/IEEE International Conference on Formal Methods and Models for Codesign, Portland, OR, Sponsored: SIGDA, SIGBED, Contact: Fei Xie, Email: xie@cs.pdx.edu

October 21–23
The 15th International ACM SIGACCESS Conference on Computers and Accessibility, Bellevue, WA, Sponsored: SIGACCESS, Contact: Clayton Lewis, Email: clayton.lewis@colorado.edu, Phone: 303-492-1592

October 21–25
ACM Multimedia Conference, Barcelona, Spain, Sponsored: SIGMM, Contact: Niculae Sebe, Email: sebe@disi.unitn.it

October 23–25
Internet Measurement Conference, Barcelona, Spain, Sponsored: SIGCOMM, SIGMETRICS, Contact: Konstantina Papgiannaki, Email: dina@tid.es

October 25–31
Conference on Systems, Programming, and Applications: Software for Humanity, Indianapolis, IN, Sponsored: SIGPLAN, Contact: Antony L. Hosking, Email: hosking@cs.purdue.edu, Phone: 765-494-6001

October 26–31
Conference on Systems, Programming, and Applications: Software for Humanity, Indianapolis, IN, Sponsored: SIGPLAN, Contact: Antony L. Hosking, Email: antony.hosking@gmail.com

October 26–28
Generative Programming: Concepts and Experiences, Indianapolis, IN, Sponsored: SIGPLAN, Contact: Jaakko Jarvi, Email: jarvi@cs.tamu.edu

stance in a larger framework. So Thinking is more aligned with higher-level abstractions and global knowledge. Thinking encourages understanding-in-the-large and this usually makes for better systems design.

Faster = Slower?

As well as encouraging unplanned experimenting, faster development tools may paradoxically increase the time to complete programming tasks. They may do this by encouraging a large number of small iterations where the learning increment is minimal, instead of a small number of thoughtful iterations where we learn a lot. So if faster might be slower, would slower speed things up?

There are two forces at work here. One is the cost of experimenting, the other is the cost of all those iterations. As development tool speed increases, clearly it costs less and less to play with a problem. This is shown in the accompanying figure by the blue bars. When the turnaround time is very long, it is usually too expensive to consider experimenting unless Thinking has come to a complete dead end. As turnaround time drops it is likely there is a threshold where Experimenting becomes much more attractive. The threshold is probably related to the perceived effort required using the two approaches.

At the same time, all the iterations generated by Experimenting have a cost that is rising gradually (shown by the red bars). With 100% Thinking, there may only be a single iteration that proves the thinking is correct. As Experimenting becomes less costly and more impromptu many of the experimental iterations may either eliminate specious dead-ends or may not expose any new knowledge. So while the iterations might not cost much, they may also be low-value in terms of learning. And we tend to do a lot of them.

Faster development tools may paradoxically increase the time to complete programming tasks.

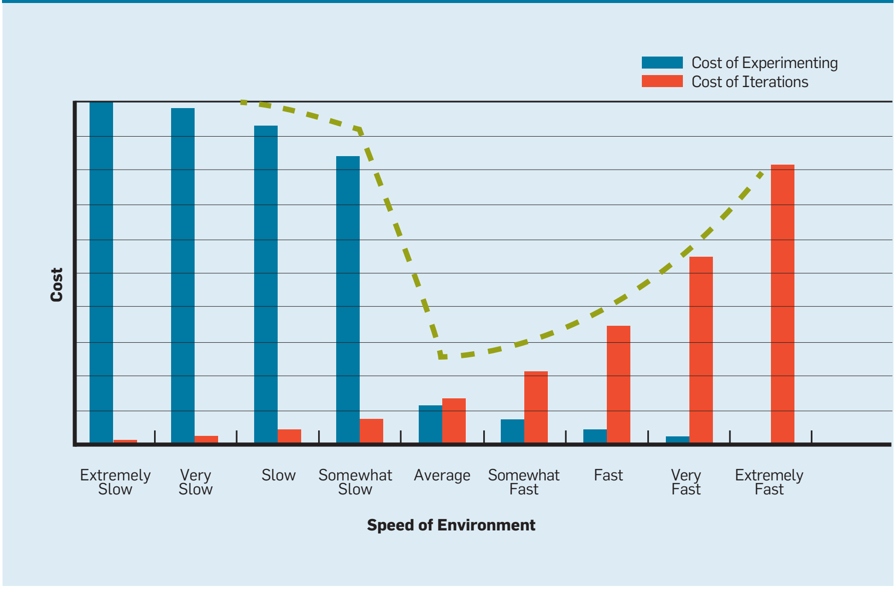
When Thinking departs the scene, Experimenting turns into hacking. Here the programmer may not even have an idea of what to look for (or even what he or she is looking *at*), not having thought through the problem at all. In this case, provided the system does not actually crash, an iteration might be considered "successful"—after all, it did "work." But we do not learn much from it.

This hypothesis infers there might be a "sweet spot" where development tools are fast, but not *too* fast—not so fast that they encourage laziness. Not so fast they encourage writing programs rather than designing programs and running programs rather than thinking about programs. This raises some questions:

- Should we consider *deliberately* introducing delays? Should the delays be on the order of seconds? Minutes? Hours?
- Would there be different kinds of delays for syntax issues versus operational or logical issues?
- Should the delay be based on the experience level of the programmer? If so, how does experience affect the delay?

There is only one criterion that should determine how we should experiment and how much we should think: *Which approach will allow us to learn most efficiently?*

The cost of experimenting versus the cost of iterations.



Effort and Learning

The putative goal of systems development is the production of code that "works" just as the putative goal of a college education is to obtain a degree. These obvious targets mask the true purpose of both activities which is to *learn*. We can print up a degree certificate in minutes and we can create code pretty much as quickly as we can type. Creating code that does what is needed requires first learning what is needed and learning is always an active and effortful endeavor. If powerful tools allow us to coast at a low energy state they can shortcut the learning process altogether. This is true of programmers, of college students, and of college students studying programming.

Computer Game

I have sometimes thought the manual computer simulation we played with pieces of paper, pretending to be a PLAN assembly language program, was so interesting and informative that it could be made into a fun board game.

Just kidding.

Phillip G. Armour (armour@corvusintl.com) is a senior consultant at Corvus International Inc., Deer Park, IL, and a consultant at QSM Inc., McLean, VA.

The idea for this column came from a series of discussions with Dave Berque (dberque@depauw.edu), professor and chair of computer science at DePauw University, to whom I would like to express my thanks.

DOI:10.1145/2505339

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
The Naming of Hosts Is a Difficult Matter

Also, the perils of premature rebooting.

Dear KV,
An argument recently broke out between two factions of our systems administration team concerning the naming of our next set of hosts. One faction wants to name machines after services, with each host having a numeric suffix, and the other wants to continue our current scheme of each host having a unique name, without a numeric string. We now have so many hosts that any unique name is getting quite long—and is annoying to type. A compromise was recently suggested whereby each host could have two names in our internal DNS (Domain Name System), but this seems overly complicated. How do you decide on a host-naming scheme?

Anonymous

Dear Anonymous,
I refer you to T.S. Eliot, who pointed out—sort of:

The Naming of Hosts is a difficult matter,
It isn't just one of your holiday games;
You may think at first I'm as mad
as a hatter
When I tell you, a host must have
THREE DIFFERENT NAMES.

"The Naming of Cats" (not hosts) is a poem in T.S. Eliot's poetry book, *Old Possum's Book of Practical Cats*, and its stage adaptation is Andrew Lloyd Webber's popular musical, *Cats*. The poem describes to humans how cats get their names. I took some liberties with Eliot's wording—as others have done before me—and extended the analogy to describe the naming of hosts. But given that T.S. Eliot died just about the time the first minicomputers were being designed, I do not think he had host names in mind when he wrote his poem. And that is a good thing, because if you think two names are bad, three would only be worse!

Giving something a name has a simple purpose: to make it understandable to a community of people.

The naming of hosts is a difficult matter that ranks with coding style, editor choice, and language preference in the pantheon of things computer people fight about that do not matter to anyone else in the whole world. What is even more annoying—or amusing—but actually annoying, is that if you are in the wrong bar at the wrong time, you will hear systems administrators fighting about naming schemes and crying in their drinks over the names they lovingly gave to hosts at their previous companies.

Giving something a name has a simple purpose: to make it understandable and memorable to a community of people. Naming your variables *foo*, *bar*, and *baz* is amusing in a short example program, but you would not want to maintain 100 lines of code written like that. The same is true of host names. Hosts have names because people need to know how to get to them—either to use their services or to maintain them, or both. If people were not involved, hosts could simply be identified by their Internet addresses. Unfortunately, host naming is an instance where geeks like to get creative. Even more unfortunately, geeks do not always know the difference between creative and annoying. It is all very well to decide your hosts should be named after *Star Trek*, *Star Wars*, or Tolkien or *Twilight* characters. With Tolkien you can probably write—and someone has probably already done so—a script to generate new names based on his works, just in case *The Hobbit*, *The Lord of the Rings* trilogy, and *The Silmarillion* did not have enough ridiculous names in them to begin with!

Everyone has a naming horror story. My first was at a university where the hosts were named after rivers. That would have been fine if you could remember how to spell "Seine," but once you run out of nice short names, you get to "Mississippi" and "Dnjeper." That is what I want to do when I remotely log in to a host, I want to think in my head, "*M-I* crooked letter crooked letter *I* crooked letter crooked letter *I* hump back hump back *I*," which is how I and many other American schoolchildren learned to spell Mississippi. I could go on and on about this, but then I would sound like those systems administrators I mentioned who were lamenting past host names. Here, therefore, is a short guide to picking host names.

A name you are going to use on a daily basis must be easy to type. That means no silent letters, such as in *Dnjeper*, and nothing that is too long, like *thisisthehostthatjackbuilt*.

It is a good idea to choose names that everyone you work with can pronounce. With globalization, finding pronounceable names has become more difficult, since some people cannot pick up *L* vs. *R*, or understand whether you just used a double o or a single o, and diphthongs will kill you (no, diphthongs are not a new Brazilian bathing suit). The main point here is to avoid picking a name with a lot of sounds that are difficult to translate into typing. Typing is still faster than using a voice-recognition system; so remember, these names will have to be typed.

If you are going to use services as names, make sure you can replace the systems behind the names without hiccups. It should be obvious that everyone is going to be annoyed if they have to use mail2.yourdomain.com when mail.yourdomain.com goes down. (This point is not really about naming, because any systems administrator worth his or her paycheck can build a system like this; but I have seen it done the wrong way, so I wanted to state it for the record.)

Avoid at all costs having two different, unrelated names for the same thing. In fact, this is true in code and host names. If you have two similar services and you want two different names, make it completely obvious how to map one name to the other and back. It is maddening to have the kind of back and forth where one person asks,

"Hey, can I reboot *fibble*?"

"Yes."

And then someone asks,

"Who rebooted *mail1*?"

"But I didn't know it was *mail1*; I thought it was *fibble*."

Finally, try to avoid being cute. I know that giving this piece of advice is basically tilting at windmills, but I have to say that people who name their mail servers male and female make my normally icy blood boil.

KV

Dear KV,

One of my company's frontline engineers—in the group that looks at the live traffic hitting our switches and servers—keeps reporting problems, and then, before anyone can look at the server that is having issues, reboots the system to clear the problem. How do you explain to someone there is information that needs to be collected when the system is misbehaving that is absolutely vital to finding and solving the problem?

Booted

Dear Booted,

I would start by standing with my foot on this person's chest and yelling, "There is information that needs to be collected when the system is misbehaving that is absolutely vital to finding and solving the problem." But I take it you have tried that already, though perhaps without enough screaming.

True, systems tend to build up state during execution that is not written to some permanent storage often enough. The problem you need to solve is not preventing the person from insta-booting a misbehaving machine, as much as it is to make sure there is a good, searchable record of what the system is doing when it is running. Most system-monitoring tools on modern servers generate plain text output. It is a simple matter to write scripts that execute periodically to write the output of these tools—such as procstat, netstat, iostat, and the like—into files that will be preserved across reboots.

For more pernicious problems, you can write your own tools, either scripts or new programs that are executed when the system is shut down or rebooted. In this way, if people are insta-booting your machines before you can get to them, you can make it so their reboot command does your bidding. You can even go so far as to rig your operating system to produce a kernel core dump on each reboot. This gives you a snapshot of the system as it was when it was broken, which you can go back to later and pick through. I warn you, though, that picking through a kernel core dump is about as much fun as picking fleas off a dog.

The only downside to collecting all this data is analyzing it. Since it is no longer really necessary to delete data, you may wind up spending a good deal of time organizing it into trees of trees of files. I offer a couple of quick suggestions. Do not make the tree scheme too difficult to traverse, either for a person or a program. It can take a very long time to access a ton of files in deep trees, due to the cost of traversing the directory trees. Keep things simple for both yourself and your analysis programs. Before you start, have a plan for what you want to store, where you want to store it, and how you plan to access it. Most people put this kind of thought into their applications, but not enough into how and where they store logs or other runtime information generated by their systems. You should put at least half as much time into the latter as you do into the former.

KV

Related articles on queue.acm.org

Kode Vicious Gets Dirty
George Neville-Neil
http://queue.acm.org/detail.cfm?id=1071723

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

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

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and a member 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:10.1145/2508701

Kevin Fu and James Blum

Inside Risks

Controlling for Cybersecurity Risks of Medical Device Software

Medical device hacking is a red herring. But the flaws are real.

WHILE COMPUTER-RELATED FAILURES are known to play a significant role in deaths and injuries involving medical devices reported to the U.S. Food and Drug Administration (FDA),[1] there is no similar reporting system that meaningfully captures security-related failures in medical devices.

Medical device software must satisfy system properties including safety, security, reliability, resilience, and robustness among others. This column focuses on the challenges to satisfying a security property for medical devices: post-market surveillance, integrity and availability, and regulation and standards.

Medical devices depend on software for patient care ranging from radiation therapy planning to pharmaceutical compounding to automated diagnosis of disease with mobile medical apps. Meanwhile, the medical community has observed an uptick in reported security vulnerabilities in medical device software—raising doubts of cybersecurity preparedness. It should come as little surprise that security risks in medical devices "could lead to patient harm" as recently explained by the chief scientist at the U.S. Food & Drug Administration's Center for Devices and Radiological Health.[6] Device manufacturers and healthcare providers ought to more carefully and deliberately consider security hazards during the phases from design to use of medical devices.

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

Measuring Medical Device Security: Quantitative or Qualitative?

Between 2006 and 2011, 5,294 recalls and approximately 1.2 million adverse events of medical devices were reported to the FDA's Manufacturer and User Facility Device Experience (MAUDE) database.[1] Almost 23% of these recalls were due to computer-related failures, of which approximately 94% presented medium to high risk of severe health consequences (such as serious injury or death) to patients.[1] For security incidents on medical devices, no systematic national reporting system exists.[3] Yet, individual hospitals know of hundreds of security incidents on medical devices.[6]

For instance, the FDA MAUDE does not capture adverse events such as lack of or impaired availability of function when malware infects a medical device's operating system. The FDA's own disclaimer explains that the MAUDE database is qualitative rather than quantitative. MAUDE is incomplete with underreporting and reporting bias.

Imagine the reaction of a clinician using a high-risk pregnancy monitor that begins to perform more slowly because of a Conficker infection. Would the clinician report a malware infection? Likely not. Admitting to playing a role in accidentally infecting a medical device would likely lead to consequences ranging from disciplinary action to loss of reputation. Thus, the actual incidence of security failures leading to healthcare delivery failures may be significantly greater than the available statistics suggest. To have better understanding of medical device security, the bad-news diode must be shorted. Reporting must be incentivized rather than penalized.

Consequences of Cybersecurity Unpreparedness for Medical Devices: Integrity and Availability

If you watch television crime dramas, you may be duped into thinking that hacking of medical devices is the number-one risk for public health today. You would be wrong. The most pressing risks are much less sexy: the unavailability of patient care and the lack of health data integrity. Here, we highlight a few examples that illustrate the consequences of unavailability and lack of integrity.

Availability of software to deliver safe and effective patient care. Interventional radiology suites and cardiac catheterization labs contain a number of computer systems to perform time-sensitive cardiac procedures, such as angioplasty, to open blocked arteries for improved outcomes in patients suffering acute heart attacks or strokes.

According to the *Wall Street Journal*,[6] a VA catheterization laboratory in New Jersey was temporarily closed in January 2010. Malware had infected the computer systems. The consequence? Patients do not receive the safe and effective care they deserve when malware causes unavailability of care. The VA has experienced hundreds of malware infections in medical devices such as X-ray machines and lab equipment made by well-known, reputable companies.

Why does old malware persist on medical devices?

Old software, old malware. Conficker was detected on 104 devices at the James A. Haley Veterans' Hospital in Tampa.[6] The affected devices included an X-ray machine, mammography, and a gamma camera for nuclear medicine studies. Conficker is a relatively old piece of malware with well-known mitigation strategies. Why does old malware persist on medical devices?

We observe that one of the cultural challenges to improved cybersecurity and therefore safety and effectiveness is a lifecycle mismatch. For instance, operating system software with production lifecycles measured in months does not match well with a medical device having production lifecycles measured in years or decades. The equivalent of a transformer for impedance matching does not yet exist for safely connecting these different production cultures.

Risks of depending on unsupported software has parallels to depending on a device where parts are no longer manufactured or repaired. Medical devices still rely on the original versions of Windows XP (circa 2001). In October 2012, the Beth Israel Deaconess Medical Center in Boston reported to the NIST Information Security and Privacy Advisory Board that the hospital depends on 664 Windows-based medical devices primarily because of supply chain issues. Of the 664 computers, 600 devices run the original version of Windows XP. There are no Service Pack 1 (SP1) machines, 15 SP2 machines, and one SP3 machine. One MRI machine still runs Windows 95. Security support for SP1 and SP2 ended on October 10, 2006 and July 13, 2010; security support for SP3 will end on April 14, 2014. In many cases, a medical device manufacturer does not provide an effective way for hospitals to upgrade to supported versions of operating systems. Today, healthcare providers are told to maintain a secure system from insecure devices.

Integrity: High-risk pregnancy monitor infected with malware. A medical device infected with malware can stray from its expected behavior. For instance, malware can cause a device to slow down and miss critical interrupts. When this happened on a high-risk pregnancy monitor, healthcare professionals could no longer trust the integrity of the sensor readings, and depended on backup methods.[4]

Availability: Anti-virus mishap disables hospital workflow. Anti-virus software can help mitigate certain cybersecurity risks, but can also introduce its own risks. On April 21, 2010, one-third of the hospitals in Rhode Island were forced to "postpone elective surgeries and stop treating patients without traumas in emergency rooms" because an automated anti-virus software update had accidentally misclassified a critical Windows DLL as malicious. The problem with anti-virus software is that by definition, anti-virus software is a post-market afterthought to make up for design flaws in the device. Anti-virus software does not remove the need to incorporate security into the early design of medical devices.

Regulation: U.S. Food and Drug Administration Actions on Cybersecurity

According to the FDA mission statement, the agency holds responsibility for protecting the public health by assuring the safety, efficacy, and *security* of medical devices. In June of this year, the FDA issued draft guidance on cybersecurity,[5] and gave examples of what FDA reviewers would expect to see during pre-market review. The draft guidance intentionally does not prescribe any particular approach or technology, but instead recommends that manufacturers consider cybersecurity starting at the concept phase of the medical device.

The FDA recommends that manufacturers provide:

- A specific list of all cybersecurity risks that were considered in the design of a device;
- A specific list and justification for all cybersecurity controls that were established for a device;
- A traceability matrix that links actual cybersecurity controls to the cybersecurity risks that were considered;
- The systematic plan for providing validated updates and patches to operating systems or medical device software, as needed, to provide up-to-date protection and to address the product lifecycle;
- Appropriate documentation to demonstrate that the device will be provided to purchasers and users free of malware; and
- Device instructions for use and product specifications related to recommended anti-virus software and/or firewall use appropriate for the environment of use, even when it is anticipated that users may use their own virus protection software.

International Role of Standards Bodies, Manufacturers, and Clinical Facilities

Standards bodies are taking actions to improve medical device cybersecurity. For instance, the Association for the Advancement of Medical Instrumentation (AAMI) recently formed a working group on medical device security that includes engineers from manufacturing and regulators. AAMI has already released standards specific to network-related cybersecurity risks (ANSI/AAMI/IEC-80001). International harmonization of cybersecurity guidance is likely on the horizon, given that phrases such as "security patches" appear in proposals from the International Medical Device Regulators Forum.

Recommendations to Improve Medical Device Cybersecurity

- Manufacturers should consider cybersecurity during the design phase of the medical device. Security is difficult to bolt on after the fact, and is most effective when designed in.
- Incentivize user facilities (for example, hospitals) to report security incidents and vulnerabilities that could lead to harm. This activity will help to gain insight into hazards that affect integrity and availability of medical devices.
- Match the production lifecycles of underlying software to the production lifecycles of the medical device. If a component is known to have a limited lifetime, then the medical device using that component runs the risk of inheriting the limited lifetime.

Conclusion

Modern healthcare delivery depends on medical device software to help patients lead more normal and healthy lives. Medical device security problems are real, but the focus on hacking goes only skin deep. Consequences of diminished integrity and availability caused by untargeted malware include the inability to deliver timely and effective patient care. By addressing security and privacy risks at the concept phase, medical devices can remain safe and effective despite the cybersecurity threats endemic to computing. Security of medical devices is more than just a potential problem on the horizon. C

References

1. Alemzadeh, H. et al. Analysis of safety-critical computer failures in medical devices. *IEEE Security and Privacy* (July–Aug. 2013), 14–26, Co-published by the IEEE Computer and Reliability Societies.
2. Fu, K. Trustworthy medical device software. In *Public Health Effectiveness of the FDA 510(k) Clearance Process: Measuring Postmarket Performance and Other Select Topics: Workshop Report*, Washington, D.C., July 2011. IOM (Institute of Medicine), National Academies Press.
3. Kramer, D.B. et al. Security and privacy qualities of medical devices: An analysis of FDA postmarket surveillance. *PLoS ONE 7*, 7 (July 2012).
4. Talbot, D. Computer viruses are "rampant" on medical devices in hospitals. *MIT Technology Review* (Oct. 17, 2012); http://www.technologyreview.com/news/429616/computer-viruses-are-rampant-on-medical-devices-in-hospitals/
5. U.S. FDA. Content of premarket submissions for management of cybersecurity in medical devices—Draft guidance for industry and Food and Drug Administration staff (June 14, 2013); http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm356186.htm.
6. Weaver, C. Patients put at risk by computer viruses. *The Wall Street Journal* (June 13, 2013); http://online.wsj.com/article/SB10001424127887324188604578543162744943762.html.

Kevin Fu (kevinfu@umich.edu) is an associate professor in the Division of Computer Science and Engineering at the University of Michigan.

James Blum (jmblum@med.umich.edu) is an assistant professor in the Department of Anesthesiology with the University of Michigan Health System.

This work was supported in part by NFS CNS-1331652 and HHS 90TR0003/01. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of NSF or HHS.

DOI:10.1145/2505340

David H. Ackley

Viewpoint Beyond Efficiency

Esteem for efficiency should be tempered with respect for robustness.

COMPUTER SCIENCE OFTEN emphasizes processing efficiency, leaving robustness to be addressed separately. However, robustness requires *redundancy*, which efficiency eliminates. For safer and more scalable computing, we must embrace and manage this trade-off.

You've seen them: Crashed computers, frozen on the job (see Figure 1). Fortunately, the result is seldom worse than user inconvenience or owner embarrassment. Still, as computer scientists we wonder why the computer inside the machine is so often the weakest link.

Computers keep gaining new responsibilities. In everything from smartphones to cars to medical equipment, we need computers to be *robust*. They should be competent at their jobs, but also sensible about the unexpected, and prudent about the malicious.

Over the years we have learned much about how to keep computers working. Fields like fault tolerance[10] and software reliability[7] employ structured redundancy to enhance robustness. Data centers and other high-availability systems have benefited, but the techniques rarely reach the mass market. Meanwhile, many areas of computer science—for example, algorithm and database design, and programming generally—view redundancy as waste. A common perspective, here stated categorically for emphasis, says software designers and programmers should assume a 100% reliable deployment platform, and the goal of software is 'CEO': *Correctness and **E**fficiency **O**nly*.

Today's crashed machines may be just canaries in the coal mine.

That 'CEO Software' mind-set has gone largely unchallenged because it has history and technology behind it. Our traditional digital architectures, error-correcting hardware, and fault-masking subsystems like TCP libraries work together to support it. Yet it is misleading and risky. It implies efficiency and robustness are separate, when actually they are coupled. Teaching it to our students perpetuates that misunderstanding.

CEO Software powers much of computing today, often with great success, but it is neither inevitable nor harmless in general. This Viewpoint reviews its origins, offers a canonical example of its hidden risks, and suggests opportunities for balancing efficiency and robustness throughout the computational stack.

It is easy to blame the woes of modern computing on flaky hardware, miscreants spreading malware, clueless users clicking, sloppy coders writing buggy code, and companies shipping first and patching later. Certainly in my programming classes I have a stern face for missing error checks and all program ugliness, but the deeper problem is our dependence on the basic von Neumann model of computation—a CPU with RAM, cranking out a vast daisy chain of ideal logical inferences. This is ultimately unscalable, as von Neumann himself observed[11] in 1948. We have stretched von Neumann's model far beyond what he saw for it. The cracks are showing.

The original concept of general-purpose digital computing amounts to this: ***Reliability** is a hardware problem; **desirability** is a software problem*. The goal was to engineer a "perfect logician" to flawlessly follow a logic 'recipe' provided later. Reliability was achieved through *massive redundancy*, using whole wires to carry individual bits, and amplifiers everywhere to squish out errors. Analog machines of the day could compute a desirable result, such as an artillery firing solution, using less than 10 amplifiers. Early digital machines did similar work more flexibly, but used thousands of amplifiers.

Meanwhile, computer scientists and programmers devised correct recipes for desirable computations. The huge costs but modest abilities of early hardware demanded those recipes be *ruthlessly efficient*. CEO Software was born. Many efficient algorithms were developed, along with efficiency enhancements such as keeping the processor busy with multiple recipes, sharing memory to speed task interactions, and caching intermediate results to save time.

However, if an intermediate result might—for any reason—be *incorrect*, reusing it might just make things worse. Sharing resources lets problems cascade between recipes. If efficiency is optimized, a single fault could corrupt a machine's behavior *indefinitely*—an unlikely outcome with random faults, but a fact of life with hostile security faults.

Computers used to have paid staff watching them. Software was small enough to debug credibly. Computing needs changed slowly. Leaving reliability to hardware worked well. The world has changed: In the hurly-burly of networked consumer computing, the more CEO Software we add to a system, the *more breakable* it gets. Today's crashed machines may be just canaries in the coal mine.

To see how robustness and efficiency can trade off, consider sorting a list by comparing pairs of items. This is considered a solved problem in computer science, with solid theory and many efficient algorithms. "Quick sort" and "merge sort" are particularly widely used. When sorting a long list, either one blows the doors off "bubble sort," which basically makes many passes over the list and swaps adjacent items if they compare out of order. Computer scientists love to hate bubble sort, because it is inefficient but easy to reinvent. It keeps popping up in software like a weed.

Efficient sorting algorithms make comparisons that often move items long distances. Bubble sort does not do that. It compares items repeatedly. It only moves items one position at a time. Bubble sorting a billion items would involve a billion billion comparisons and take years. *Quick sorting* a billion items took under 10 minutes on my laptop: It is an *efficient* algorithm for big tasks.

Yet some tasks are small, like sorting my shopping list by price. Even inefficient bubble sort will do that in the blink of an eye. Is it always best to do even small jobs as fast as possible? Maybe not.

As a demonstration, imagine sorting just 52 items, like a shuffled deck of cards. I used[1] stock implementations of quick and merge sorts, and hand-coded a totally unoptimized bubble sort. Each algorithm picks pairs of cards to compare, and passes them to a 'card comparison component' to determine their order.

Here's the twist: Imagine that component is *unreliable*. Maybe malware corrupted it. Maybe sunspots cooked it. We don't know why. For this demonstration, the card comparison component usually works fine, but on 10% of comparisons it gives a random answer. It might claim 3♠ > 7♢, or Q♡ = 9♣, or whatever.

Given such faults, a sorting algorithm's output might well be incorrect. I scored the output error by adding up each card's distance from its proper position—ranging from 0, for the correct deck order, up to 1,352 (that is, $52^2/2$), for worst cases like getting the whole deck backward.

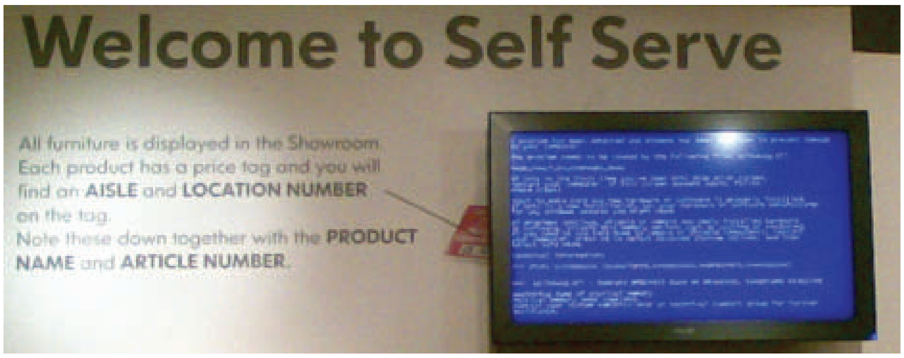
Averaged over 1,000 shuffled decks, the errors made by each sorting algorithm are shown in Figure 2.

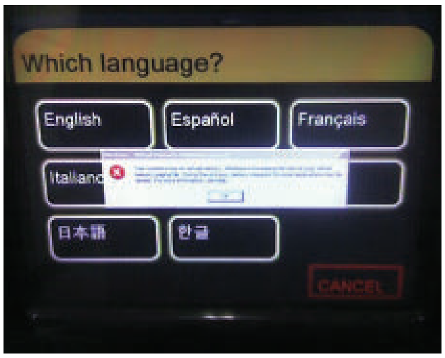
Now whose doors have fallen off? Bubble sort's inefficient repeated comparisons repair many faults, and its inefficient short moves minimize the damage of the rest.

```
c:\diebold\css\css.exe
C:\Diebold\EXE\chprdrh.exe
CHPRDRH: The DEVICE.INI setting trace could not be found.
CHPRDRH: The terminal type could not be determined from configuration.
CHPRDRH - DIP Smart Card Reader. Version 30010
Copyright (C) Diebold 2003. All rights reserved.

CHPRDRH: Port COM2 opened.
CHPRDRH: Baud rate at 38400.
CHPRDRH: Line Control at 7 bits. Even parity. 1 Stop bit.
CHPRDRH: Device Control Block set.
CHPRDRH: DCB Received 2nd time.
CHPRDRH: Initializing DIP SMART CARD driver.
CHPRDRH: Initializing DIP SMART CARD reader.
CHPRDRH: Initializing CSS Connection.
CHPRDRH: CSS Error attaching to CHPAPWIFC (51), retrying....
CHPRDRH: Successfully established CSS connection.
CHPRDRH: Successfully established CSS connection.
```
Recycle Bin
Do you want to turn off your Active Desktop? If so,
Right-click the desktop, point to Active Desktop, and then
click Show Web Content
Start
Cancel
Clear

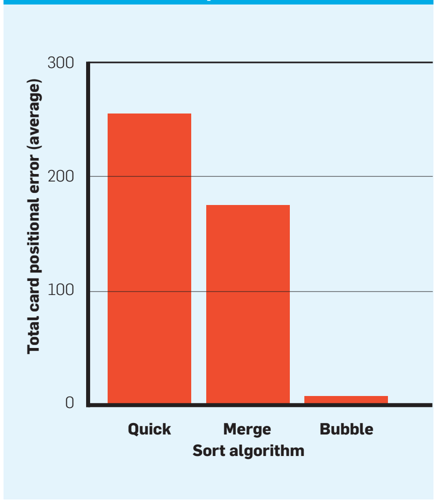
Figure 1. Crashed computers are a common sight.


Welcome to Self Serve
All furniture is displayed in the Showroom
Each product has a price tag and you will
find an AISLE and LOCATION NUMBER
on the tag.
Note these down together with the PRODUCT
NAME and ARTICLE NUMBER.



PHOTOGRAPHS BY DEEPAK SARDA, CLIVE DARRA, LASER BURNERS

Figure 2. Errors sorting a deck of cards with 10% failed comparisons.



I subsequently found that quick and merge become competitive with bubble if I repeat each of their comparisons six times and return the most common answer. But that felt like a cheap hack—an after-the-fact fix tailored to one situation. Here, the failed comparisons were *independent, identically distributed* (IID) events, but real-world faults often occur in bursts or other patterns. For example, a mistaken person gives wrong directions even if you repeat the question. How far should you drive based on their answers? Or in this case, how far should a sorter swap data?

Bubble sort wins on some non-IID faults as well. Suppose our fallible comparator also suffered from a sticky output, so half the time it just repeated its previous answer. Given such awful directions, bubble's error rose to nearly 90—but quick exceeded 550, merge exceeded 650, and even their best-of-six variants both broke 250. Bubble sort is inefficient, but it is *robust*.

Efficiency is key when demands exceed resources. Yet many computers are usually idling. They could have been checking their work, but that is off the radar for CEO Software. Optimizing efficiency conspires with program correctness to undermine robustness. The more work done per program action, as with the long swaps of efficient sorters, the more serious the disruption from faulty action. Yet as long as the optimized code remains logically correct, with CEO Software there is no pushback. That risk is never even assessed, because fault-induced failures, no matter how severe, are billed to someone else.

Ironically, even fault tolerance researchers can succumb to the lure of CEO Software. With commendable candor, one group has argued that optimizing fault-free efficiency, "a practice that we have engaged in with relish..., is increasingly misguided, dangerous, and even futile."[6] Broader awareness of CEO Software's liabilities could help avoid such traps.

Over 60 years ago, von Neumann recognized the scalability limits of his namesake design. He predicted a future computing model, a "logic of automata," in which short runtimes would be preferred, and all computational operations would tolerate faults.[11] With transistor sizes continuing to shrink beneath churning piles of dubious software, plus fault-injecting attackers wielding lasers and X-rays,[3] we continue to delay that future at our peril.

Today's computing base is optimized for the perfect logician, so the quest for robustness has many fronts. A clean-slate approach is robust, indefinitely scalable architectures,[2] but research topics abound.

For example, it would help to have better visibility into efficiency-robustness trade-offs. Existing software quality metrics could be expanded to include correctness degradation under various fault models. Bubble sort was not even designed to be robust. How will other algorithms behave?

To set the stage formally, it would help to have results on the trade-offs between efficiency and robustness, perhaps like a Brewer's Theorem[8] extended to degrees of failure. A starting point might be *highly optimized tolerance*,[4] which explicitly represents failure size.

We also need to understand robust applications programming better. The ongoing trend away from monolithic mainlines toward quick event handlers could be seen as compatible with von Neumann's short program runtimes.

More possibilities arise as we replace prima donna CEO Software with robust team player code that gives better than it gets. For example, Dijkstra's *self-stabilization*[9] technique continually "falls toward" correctness. It is a beautiful embodiment of the robust spirit, but requires extra work to interface with CEO Software, because it can violate correctness while stabilizing, which CEO Software cannot tolerate.

Finally, we could develop self-tuning algorithms that trade efficiency against robustness dynamically, based on workload properties, and resource availability and quality. Some existing work[5] uses failure-detection heuristics to switch from efficient to robust processing. More could be done in a computational model that supported signaling external resource characteristics, and internal robustness margins, between components.

It is time to study and manage incorrectness in the interest of robustness. We should not shun the trade-off, but rather, we should understand, engineer, and teach computation beyond correctness and efficiency only.

Robust-first computation now. C

References

1. Ackley, D.H. Bubble sort robustness demonstration code (Apr. 2012); http://livingcomputation.com/robusort2.tar.
2. Ackley, D.H., Cannon, D.C., and Williams, L.R. A movable architecture for robust spatial computing. *The Computer Journal*, 2012.
3. H. Bar-El, et al. The sorcerer's apprentice guide to fault attacks. In *Proceedings of the IEEE 94*, 2 (Feb. 2006), 370–382.
4. Carlson, J.M. and Doyle, J. Highly optimized tolerance: A mechanism for power laws in designed systems. *Phys. Rev. E. 60*, (Aug. 1999), 1412–1427.
5. Chang, I., Hiltunen, M.A., and Schlichting, R.D. Affordable fault tolerance through adaptation. In *Proceedings of PPS/SPDP Workshops*, (1998), 585–603.
6. Clement, A. et al. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*. USENIX Association, Berkeley, CA, 2009, 153–168.
7. Dohi, T. and Cukic, B., Eds. *Proceedings of the IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE 2011)* (Nov. 29–Dec. 2, 2011, Hiroshima, Japan), IEEE, 2011.
8. Gilbert, S. and Lynch, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News 33*, 2 (June 2002), 51–59.
9. Schneider, M. Self-stabilization. *ACM Comput. Surv. 25*, 1 (Mar. 1993), 45–67.
10. Swarz, R.S., Koopman, P., and Cukier, M. Eds. *IEEE/IFIP International Conference on Dependable Systems and Networks*, (Boston, MA, June 25–28, 2012). IEEE Computer Society, 2012.
11. von Neumann, J. The general and logical theory of automata. In L.A. Jeffress, Ed. *Cerebral Mechanisms in Behaviour: The Hixon Symposium (1948)* Wiley, 1951, 15–19. Also appears as pages 302–306 in A.H. Taub, Ed., *John von Neumann Collected Works: Volume V—Design of Computers, Theory of Automata and Numerical Analysis*, Pergamon Press, 1963.

David H. Ackley (ackley@cs.unm.edu) is an associate professor of computer science at the University of New Mexico in Albuquerque, NM.

The author thanks Brian Christian, Howard Shrobe, George Furnas, Jim Hollan, Michael Lesk, Terry Jones, and John King for their review comments.

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

DOI:10.1145/2507771.2507779

Article development led by acmqueue
queue.acm.org

A special section on high-frequency trading and exchange technology.

BY JACOB LOVELESS

Barbarians at the Gateways

ILLUSTRATION BY SPOOKY POOKA AT DEBUT ART

I AM A former high-frequency trader. For a few wonderful years I led a group of brilliant engineers and mathematicians, and together we traded the electronic marketplaces and pushed systems to the edge of capability.

High-frequency trading (HFT) systems operate and evolve at astounding speeds. Moore's Law is of little comfort when compared with the exponential increase in market-data rates and the logarithmic decay in demanded latency. As an example, during a period of six months the requirement for a functional trading system went from a "tick-to-trade" latency of 250 microseconds to 50. To put that in perspective, 50 microseconds is the access latency for a modern solid-state drive.

I am also a former and current developer of exchange technology. The *exchange* is the focal point of HFT, where electronic buyers and sellers match in a complex web of systems and networks to set the price for assets around the world. I would argue the computational challenges of developing and maintaining a competitive advantage in the exchange business are among the most difficult in computer science, and specifically systems programming. To give you a feeling of scale, our current exchange technology is benchmarked in nightly builds to run a series of simulated market data feeds at one million messages per second, as a unit test. There is no such thing as premature optimization in exchange development, as every cycle counts.

The goal of this article is to introduce the problems on both sides of the wire.

Today a big Wall Street trader is more likely to have a Ph.D. from Caltech or MIT than an MBA from Harvard or Yale. The reality is that automated trading is the new marketplace, accounting for an estimated 77% of the volume of transactions in the U.K. market and 73% in the U.S. market.[a] As a community, it is starting to push the limits of physics. Today it is possible to buy a custom ASIC to parse market data and send executions—in 740 nanoseconds (or 0.00074 milliseconds.[4] Indeed, human reaction time to a visual stimulus is around 190 *million* nanoseconds.)

In the first of the other two articles in this special section on HFT, Sasha Stoikov and Rolf Waeber explain the role of one-pass algorithms in finance. These algorithms are used throughout the industry as they provide a simple and very fast way of calculating useful statistics (such as correlation between two streams). One-pass algorithms are also easier to implement in hardware (using Verilog or VHDL) because they require only a few bits of memory, compared with a vector of historical events.

In the other article, Stephen Strowes discusses a method for estimating round-trip time (RTT) latency from packet headers. Estimating RTT is a key component for both HFT firms and exchanges, and the method presented here solves an interesting problem when you cannot install a tap on every interface or on the other side of the wire.

My Time in HFT

When I began in HFT, it was a very different world. In 2003, HFT was still in its infancy outside of U.S. equities, which had two years earlier been regulated into *decimalization*, requiring stock exchanges to quote stock prices in decimals instead of fractions. This decimalization of the exchanges changed the minimum stock tick size

a http://bit.ly/13WpmE2

(minimum price increment) from 1/16th of a dollar to \$0.01 per share. What this meant was that "overnight the minimum spread a market-maker (someone who electronically offered to both buy and sell a security) stood to pocket between a bid and offer was compressed from 6.25 cents...down to a penny."[5]

This led to an explosion in revenue for U.S. equity-based HFT firms, as they were the only shops capable of operating at such small incremental margins through the execution of massive volume. Like the plot of *Superman III*, HFT shops could take over market-making in U.S. equities by collecting pennies (or fractions of a penny) millions of times a day. I was not trading stocks, however; I was trading futures and bonds. Tucked inside a large Wall Street partnership, I was tackling markets that were electronic but outside the purview of the average algorithmic shop. This is important as it meant we could start with a tractable goal: build an automated market-making system that executes trades in under 10 milliseconds on a 3.2GHz Xeon (130nm). By 2004, this was halved to five milliseconds, and we were armed with a 3.6GHz Nocona. By 2005 we were approaching the one-millisecond barrier for latency arbitrage and were well into the overclocking world. I remember bricking a brand-new HP server in an attempt to break the 4.1GHz barrier under air.

By 2005, most shops were also modifying kernels and/or running real-time kernels. I left HFT in late 2005 and returned in 2009, only to discover the world was approaching absurdity: by 2009 we were required to operate well below the one-millisecond barrier, and were looking at tick-to-trade requirements of 250 microseconds. *Tick to trade* is the time it takes to:

1. Receive a packet at the network interface;
2. Process the packet and run through the business logic of trading;
3. Send a trade packet back out on the network interface.

To do this, we used real-time kernels and in my shop, we had begun implementing functionality on the switches themselves (the Arista switch was Linux based, and we had root access). We must not have been alone in implementing custom code on the switch, because shortly after, Arista made a 24-port switch with a built-in field-programmable gate array (FPGA).[1] FPGAs were becoming more common in trading—especially in dealing with the increasing onslaught of market-data processing.

As with all great technology, using it became easier over time, allowing more and more complicated systems to be built. By 2010, the barriers to entry into HFT began to fall as many of the more esoteric technologies developed over the previous few years became commercially available. Strategy development, or the big-data problem of analyzing market data, was a great example. Hadoop was not common in many HFT shops, but the influx of talent in distributed data mining meant a number of products were becoming more available. Software companies (often started by former HFT traders) were now offering amazing solutions for messaging, market-data capture, and networking. Perhaps as a result of the inevitable lowering of the barriers to entry, HFT was measurably more difficult by 2010. Most of our models at that time were running at half-lives of three to six months.

I remember coming home late one night, and my mother, a math teacher, asked why I was so depressed and exhausted. I said, "Imagine every day you have to figure out a small part of the world. You develop fantastic machines, which can measure everything, and you deploy them to track an object

Figure 1. Using NAT to conform to the exchange transit network.

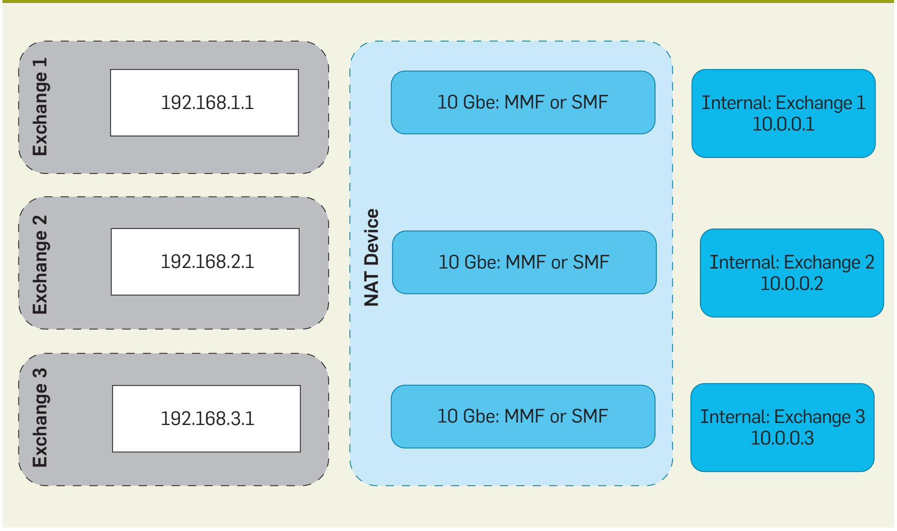
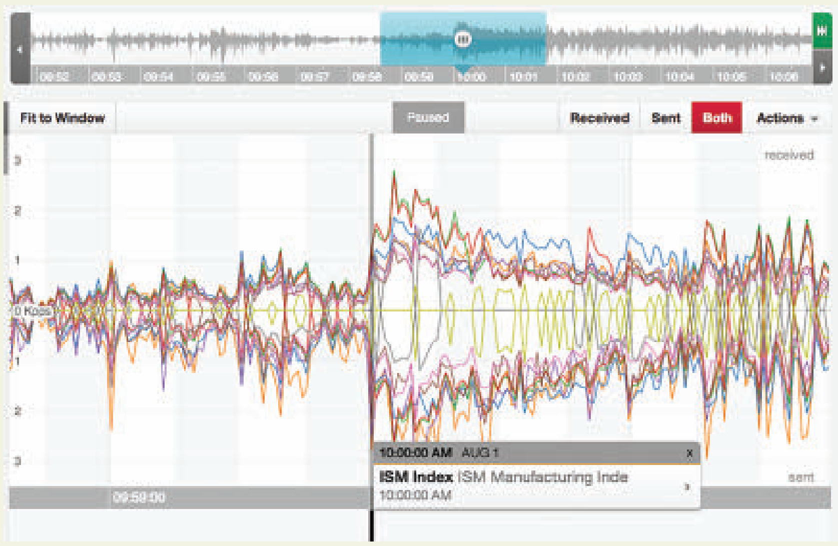


Figure 2. Monitoring a packet burst.



falling. You analyze a million occurrences of this falling event, and along with some of the greatest minds you know, you discover gravity. It's perfect: you can model it, define it, measure it, and predict it. You test it with your colleagues and say, 'I will drop this apple from my hand, and it will hit the ground in 3.2 seconds,' and it does. Then two weeks later, you go to a large conference. You drop the apple in front of the crowd… and it floats up and flies out the window. Gravity is no longer true; it was, but it is not now. That's HFT. As soon as you discover it, you have only of few weeks to capitalize on it; then you have to start all over."

The HFT Technology Stack

What follows is a high-level overview of the modern HFT stack. It is broken into components, though in a number of shops these components are encapsulated in a single piece of hardware, often as FPGA.

Collocation. The first step in HFT is to place the systems where the exchanges are. Light passing through fiber takes 49 microseconds to travel 10,000 meters, and that is all the time available in many cases. In New York, there are at least six data centers you need to collocate in to be competitive in equities. In other assets (for example, foreign exchange), you need only one or two in New York, but you also need one in London and probably one in Chicago. The problem of collocation seems straightforward:

1. Contact data center.
2. Negotiate contract.
3. Profit.

The details, however, are where the first systems problem arise. The real estate is extremely expensive, and the cost of power is an ever-crushing force on the bottom line. A 17.3-kilowatt cabinet will run $14,000 per month.[7] Assuming a modest HFT draw of 750 watts per server, 17 kilowatts can be taken by 23 servers. It's also important to ensure you get the right collocation. In many markets, the length of the cable *within the same building* is a competitive advantage. Some facilities such as the New York Stock Exchange (NYSE) data center in Mahwah, NJ, have rolls of fiber so that every cage has exactly the same length of fiber running to the exchange cages.[3]

Networking. Once the servers are collocated, they need to be connected. Traditionally this is done via two methods: data-center cross connects (single-mode or multimode fiber); and cross-data-center WAN links.

The Cross Connect and NAT. Inside the data center are multiple exchanges or market-data feeds. Each endpoint must conform to the exchange transit network. This is a simple Network Address Translation (NAT) problem and can be tackled at the switch level, as shown in Figure 1. Multiple vendors (for example, Arista[2]) offer hardware-based NAT at the port level. For some marketplaces (foreign exchange) there may be as many as 100 such NAT endpoints within a single data center.

The key to the internal NAT problem is the core nature of trading: correlated bursts. These bursts are what make HFT networks so ridiculously difficult. Figure 2 is a screenshot of an internal monitoring application that shows the packet-per-second rates for a (small) collection of symbols moving together during an ISM announcement. The Institute of Supply Management (ISM) manufacturing composite index is a diffusion index calculated from five of the 11 subcomponents of a monthly survey of purchasing managers at approximately 300 manufacturing firms in the U.S. At 14:00 EDT the ISM announcement is made, and a burst occurs. Bursts like this are common and happen multiple times a day.

As the world becomes more interconnected, and assets are more closely linked electronically, these bursts can come from anywhere. A change in U.K. employment will most certainly affect the GBP/USD rate (currency rates are like relative credit strength). That in turn affects the electronic U.S. Treasury market, which itself affects the options market (the risk-free rate in our calculations has changed). A change in op-

Figure 3. WAN links: high-throughput path and lower-throughput fast path.

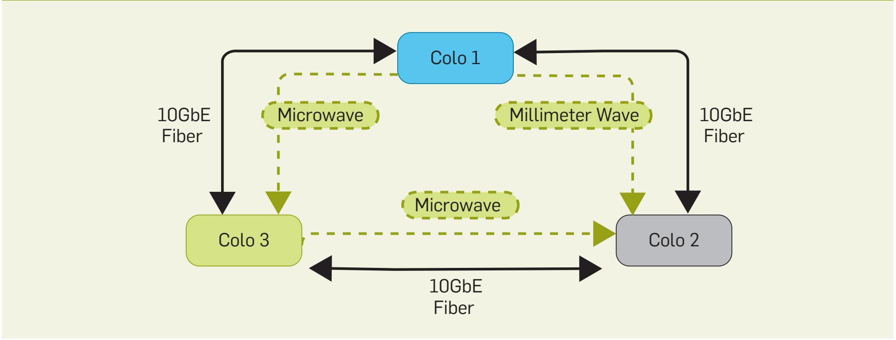


Figure 4. Feed handler parsing market-data feeds.

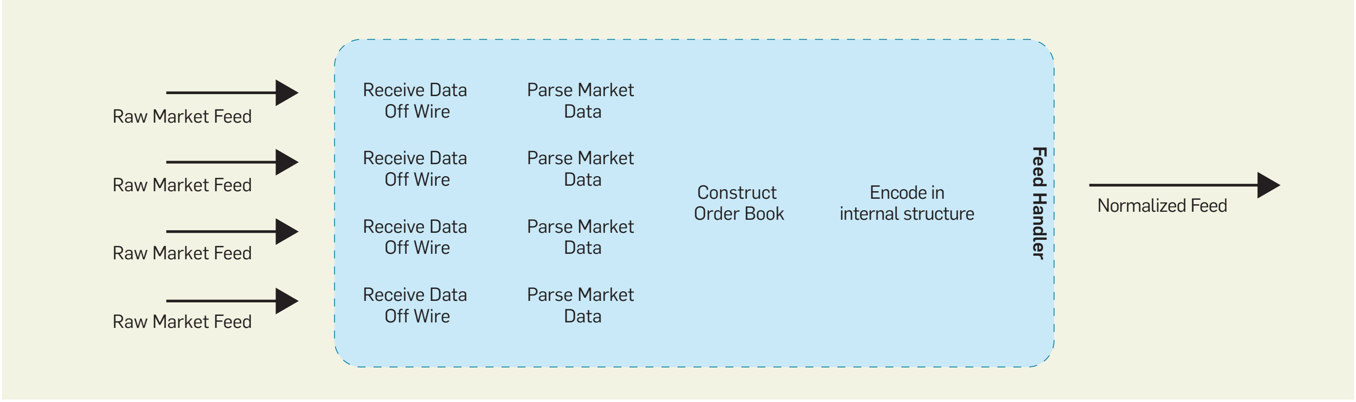


Figure 5. Tickerplant distributing market-data feeds.

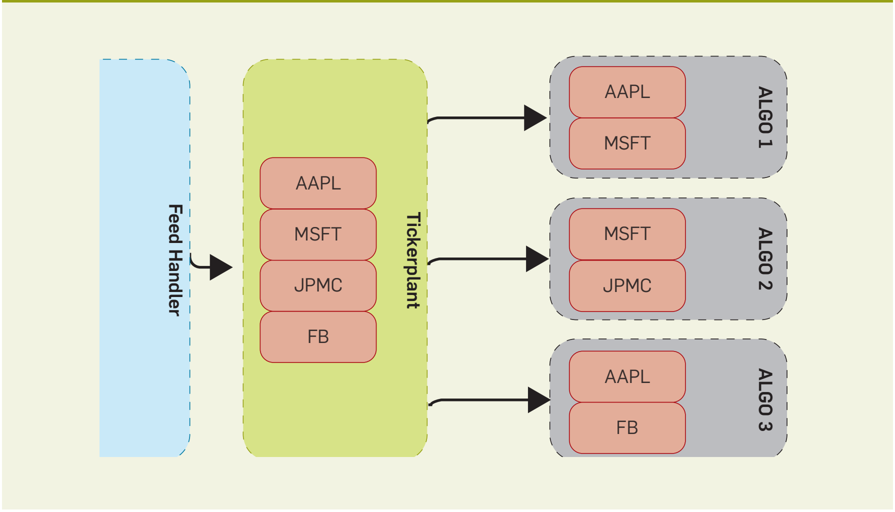


Figure 6. The order book.

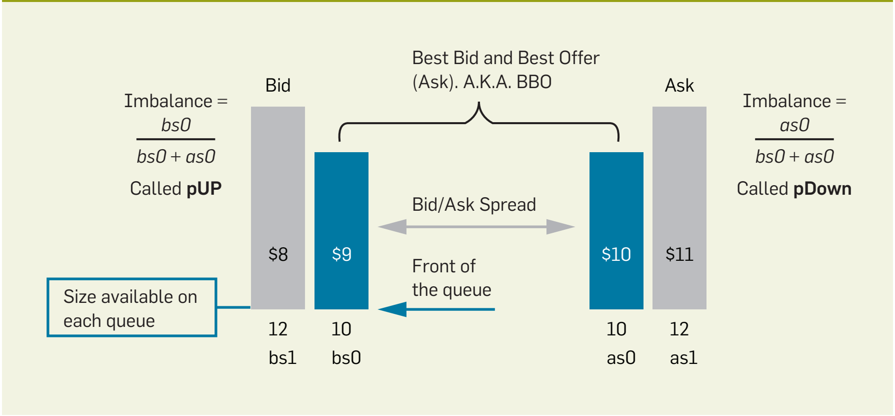
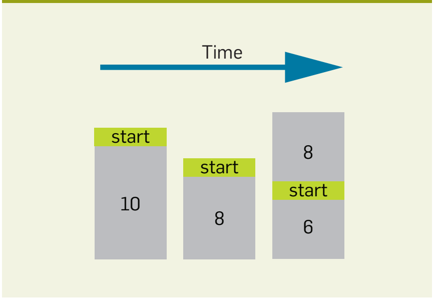

Best Bid and Best Offer (Ask). A.K.A. BBO
Bid
Ask
Imbalance = bs0 / bs0 + as0
Called pUP
Imbalance = as0 / bs0 + as0
Called pDown
Bid/Ask Spread
Front of the queue
Size available on each queue
$8
$9
$10
$11
12
10
10
12
bs1
bs0
as0
as1

Figure 7. The queue position.



tions leads to a change in large-scale exchange-traded funds (ETFs). The value of an ETF cannot be greater than the sum of its components (for example, the SPDR S&P 500), so the underlying stocks must change. Therefore, the state of employment in London will affect the price of the Dollar Tree (DLTR), which does not have a single store outside North America—it's a tangled web.

WAN Links. Outside the data center the systems need WAN links. Traditionally HFT shops ran two sets of links, as shown in Figure 3: a high-throughput path and a lower-throughput *fast path*. For the high-throughput path, private point-to-point fiber—10GbE (gigabit Ethernet) is preferred. For the fast path, each location allows for options. In the New York metro area, both millimeter and microwave solutions are available. These technologies are commonplace for HFT fast-path links, since the reduced refractive index allows for lower latency.

Feed handler. The *feed handler* is often the first bit of code to be implemented by an HFT group. As shown in Figure 4, the feed handler subscribes to a market-data feed, parses the feed, and constructs a "clean" book. This is traditionally implemented on an FPGA and has now become a commodity for the industry (http://www.exegy.com). Most feed handlers for U.S. equities are able to parse multiple market-data feeds and build a consolidated book in less than 25 microseconds.

Tickerplant. The *tickerplant* is the system component responsible for distributing the market-data feeds to the internal systems based on their subscription parameters (topic-based subscription), as shown in Figure 5. In these scenarios, the tickerplant is like a miniature version of Twitter, with multiple applications subscribing to different topics (market-data streams).

In addition to managing topic-based subscriptions, advanced tickerplants often maintain a cache of recent updates for each instrument (to catch up subscribers), calculate basic statistics (for example, the moving five-minute volume-weighted average price), and provide more complicated aggregate topics (for example, the value of an index based on the sum of the underlying 500 securities).

Low-latency applications. In my experience, most high-frequency algorithms are fairly straightforward in concept—but their success is based largely on how quickly they can interact with the marketplace and how certain you can be of the information on the wire. What follows is a simple model (circa 2005), which required a faster and faster implementation to continue to generate returns.

To begin, let's review some jargon.

Figure 8. The trading rate.

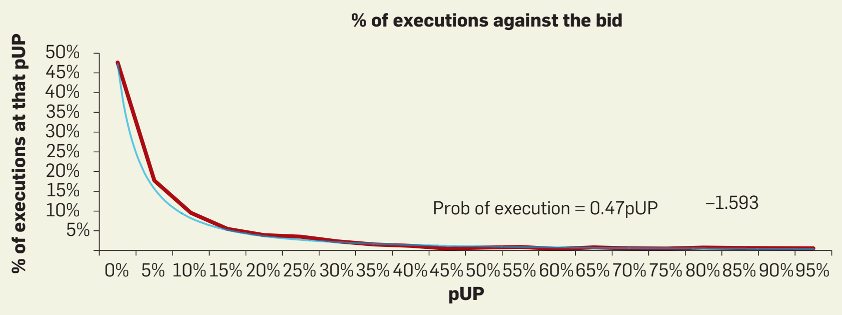


Figure 6 shows the first two levels of the *order book*. The order book is split into two sides: *bids* (prices at which people are willing to buy); and *asks* or *offers* (prices at which people are willing to sell). A *queue* consists of the individual orders within a price.

Queue Life: Getting to the Front

Orders are executed in a first-in, first-out manner (in most marketplaces). When describing the life of an individual order in an individual queue, we often say that *X* is *ahead*, and *Y* is *behind*. More generally, we say we are in the top *X*% of the queue. This is called the *queue position*. Figure 7 shows an order in the middle of the queue by the last time step: there are six shares in front of the order, and eight shares behind it. The longer an order is in the queue, the more likely it is to get to the front. The speed at which an order gets to the front is a function of two things: the rate of trading (trades take orders off the front of the queue); and the rate of cancelling of other orders.

Trading rates are somewhat difficult to estimate, but there is a clear relationship between the probability an order will get executed and the ra-

Figure 9. The empirical cancel rate.

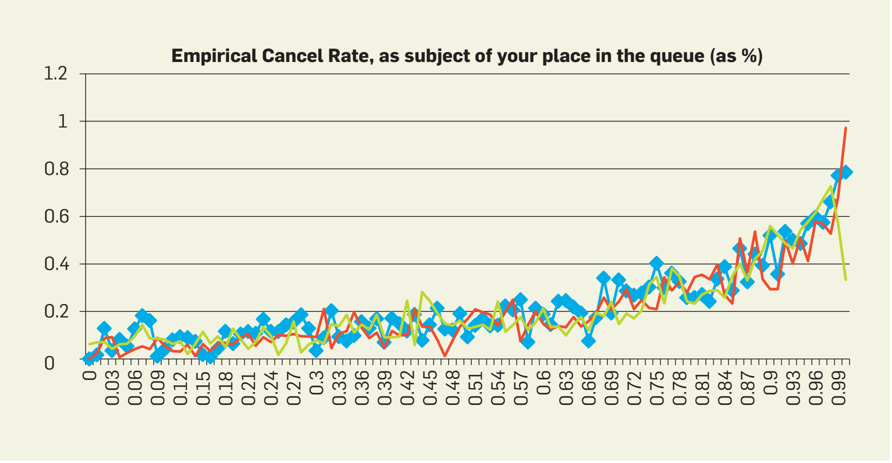


Figure 10. Two methods of getting to the front of the queue.

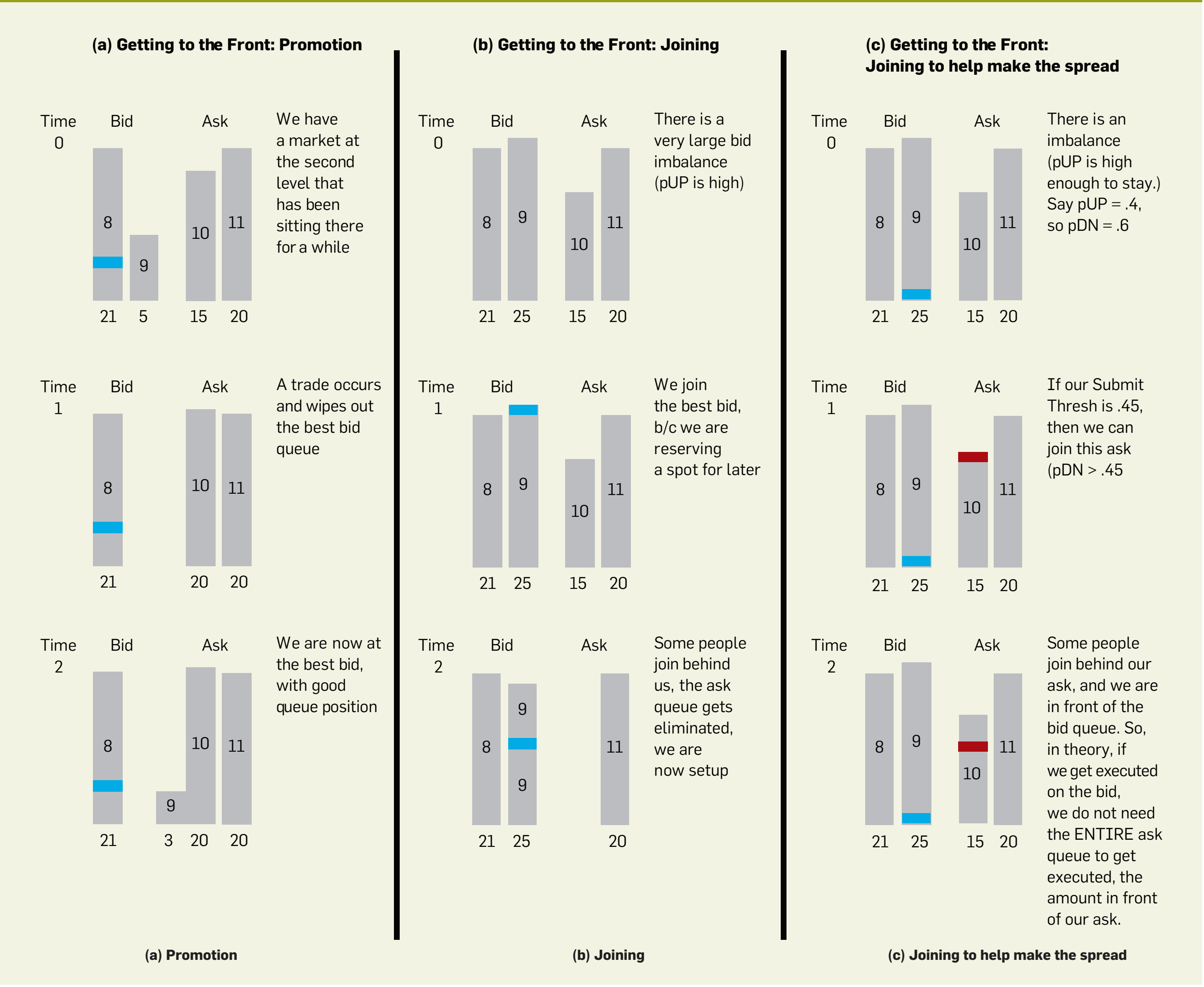
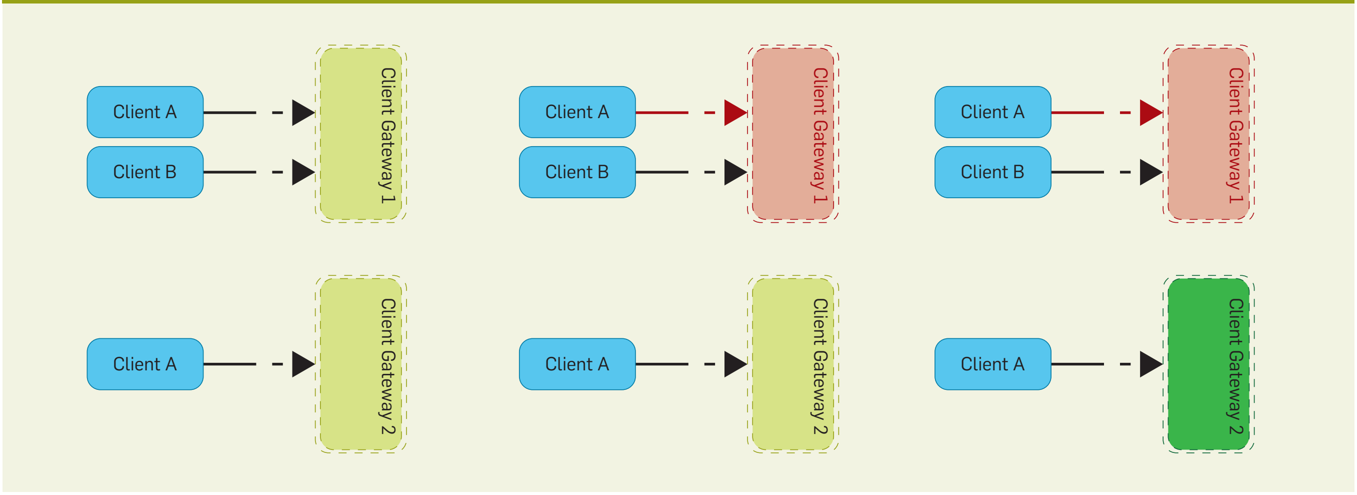

(a) Getting to the Front: Promotion
Time 0
Bid
Ask
8
9
10
11
21 5 15 20
We have a market at the second level that has been sitting there for a while
Time 1
Bid
Ask
8
10
11
21 20 20
A trade occurs and wipes out the best bid queue
Time 2
Bid
Ask
8
9
10
11
21 3 20 20
We are now at the best bid, with good queue position
(a) Promotion
(b) Getting to the Front: Joining
Time 0
Bid
Ask
8
9
10
11
21 25 15 20
There is a very large bid imbalance (pUP is high)
Time 1
Bid
Ask
8
9
10
11
21 25 15 20
We join the best bid, b/c we are reserving a spot for later
Time 2
Bid
Ask
8
9
9
11
21 25 20
Some people join behind us, the ask queue gets eliminated, we are now setup
(b) Joining
(c) Getting to the Front: Joining to help make the spread
Time 0
Bid
Ask
8
9
10
11
21 25 15 20
There is an imbalance (pUP is high enough to stay.) Say pUP = .4, so pDN = .6
Time 1
Bid
Ask
8
9
10
11
21 25 15 20
If our Submit Thresh is .45, then we can join this ask (pDN > .45
Time 2
Bid
Ask
8
9
10
11
21 25 15 20
Some people join behind our ask, and we are in front of the bid queue. So, in theory, if we get executed on the bid, we do not need the ENTIRE ask queue to get executed, the amount in front of our ask.
(c) Joining to help make the spread

Figure 11 . Gaming the gateway.



tio of its queue size versus the opposite queue (for example, the bid queue size vs. the ask queue size). Figure 8 shows this as pUP.

Cancel rates. If trading rates are difficult, cancel rates are even more difficult. The question is, given your place in the queue, what is the probability a cancel will come from in front of you (thereby allowing you to move up)? This is difficult to estimate, but our own trading and some historical data provides the basis for an engineering estimate.

To start, we know that if we are in the back of the queue, the probability of a cancel coming from in front of us is 100%. If we are in the front, it's 0%.

Figure 9 is a chart of the empirical percentage of time a cancel comes in front of your order; it is subject to whatever percentage of the queue you were in. For example, if you are at 0 on the *x*-axis, then you are at the very front of the queue, and cancels must come from behind you. The key takeaway is this: The closer your order gets to the front of the queue, the less likely an order in front of you will cancel (for example, your progression slows).

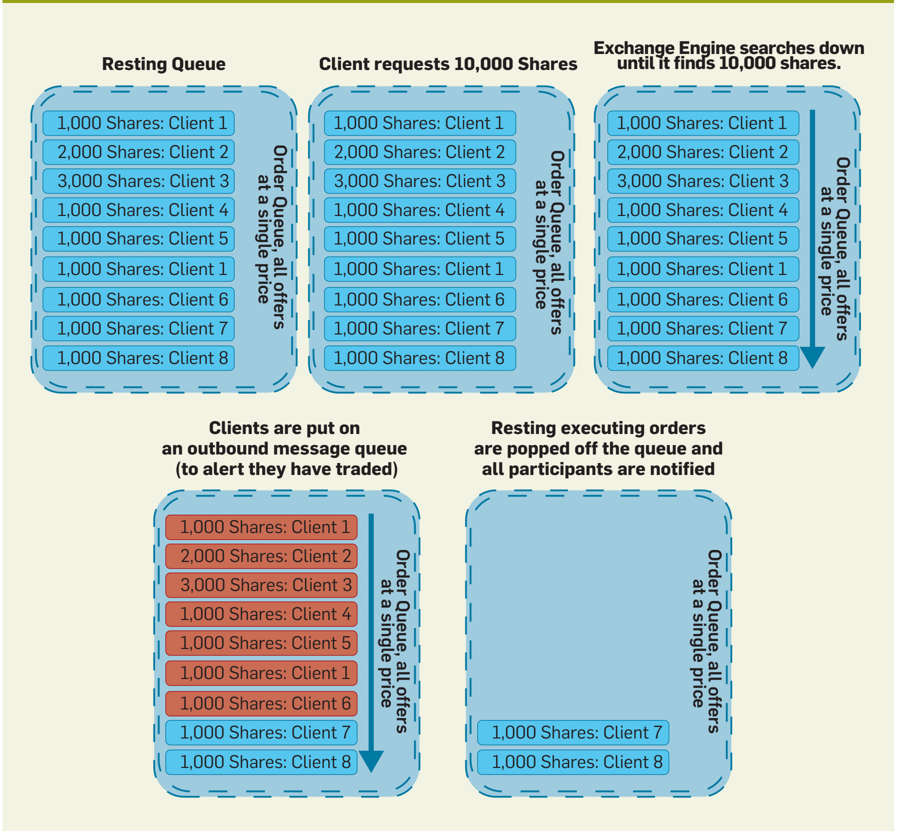
We often say you get to the front of the queue via two methods: *promotion*, which is a second-level queue becoming a first-level queue, and *joining*, which is, exactly as it sounds, joining the newly created queue. Figure 10 illustrates the difference between promotion and joining.

Profit and the big queue. If your order is executed on a large queue, then you have a free option to collect the spread. If one of your orders on the opposite queue is executed, then you collect the difference between the price at which you bought the asset (bid side of \$9) and the price at which you sold the asset (offer side of \$10).

In the event the queue you were executed on gets too small, you can *aggress* the order that was behind you. This means crossing the bid/ask spread and forcing a trade to occur. If you get executed *passively*, you are aggressed upon by another order sitting on a queue. As long as another order is behind you, you can unwind the trade, meaning you can aggress the order behind.

Aggressively unwinding a trade is called *scratching* the trade. You did not make a spread; you did not lose a spread. It is a zero-sum trade.

Figure 12. Matching engine.


Resting Queue
Client requests 10,000 Shares
Exchange Engine searches down until it finds 10,000 shares.
1,000 Shares: Client 1
2,000 Shares: Client 2
3,000 Shares: Client 3
1,000 Shares: Client 4
1,000 Shares: Client 5
1,000 Shares: Client 1
1,000 Shares: Client 6
1,000 Shares: Client 7
1,000 Shares: Client 8
Order Queue, all offers at a single price
Clients are put on an outbound message queue (to alert they have traded)
Resting executing orders are popped off the queue and all participants are notified

Exchange Technology

An exchange is the collection of systems that facilitate the electronic execution of assets in a centrally controlled and managed service. Today, exchanges are in a fight to offer faster and faster trading to their clients, facing some of the same latency issues as their newer HFT clients.

Collocation. For exchanges, collocation can be an invaluable source of revenue. The more incumbent exchanges run their own data centers (such as NYSE and NASDAQ), and customers pay collocation fees to the exchanges directly. Newer exchanges must collocate in third-party data centers that house financial customers (for example, Equinix's NY4 in Secaucus, NJ).

When exchanges operate inside third-party hosting facilities, they are subject to the same power and cooling costs as their HFT brethren. As such, many exchanges focus on delivering high-throughput systems using standard x86 designs.

Networking. Exchange networking is as challenging as HFT networking but also has a deeper focus on security. Information arbitrage, or the practice of gaining information about a market that is not directly intended for the recipient, is a major concern. An example of information arbitrage is when an exchange participant "snoops" the wire to read packets from other participants. This practice is easily thwarted with deployment of VLANs for each client.

Most exchanges still deploy 1GbE at the edge. This is both a function of history (change management in an exchange is a long process) and practicality. By putting 1GbE edge networks in place, the exchange can protect itself somewhat from the onslaught of messages by limiting both the inbound bandwidth and adding a subtle transcoding hit. For example, a 1GbE Arista 7048 has a three-µsec latency for a 64-byte frame, which is a 350-nsec latency for the same frame on a 7150 (10GbE).

Gateway. The gateway is the first exchange subsystem that client flow encounters. Gateways have evolved over the years to take on more responsibility, but at the core they serve as feed handlers and tickerplants. The gateway receives a client request to trade (historically in the FIX format, but as latency became paramount, exchanges have switched to proprietary binary protocols). It then translates the request to a more efficient internal protocol, and routes the request to the appropriate underlying exchange matching engine.

The gateway also serves as the first line of defense for erroneous trades. For example, if a client attempts to buy AAPL for $5,000 (whereas AAPL is offered at $461), the gateway can reject the order.

Market Data Gateway. Traditionally the Order Gateway (which receives client requests to trade) and the Market Data Gateway (which distributes market-data feeds) are two separate systems, and often two separate networks. For market-data distribution, two methods are common: User Datagram Protocol (UDP) Multicast for collocated customers; and Transmission Control Protocol (TCP) for noncollocated customers. Customization takes place here as well (for example, NASDAQ's SoupTCP[9]). In some markets (for example, FX), all market data is distributed over TCP in Financial Information Exchange (FIX). For the other markets, data is often distributed over UDP in a binary format or an easy-to-parse text format. The predominant two binary formats are ITCH[6] and OUCH,[8] and both sacrifice flexibility (fixed-length offsets) for speed (very simple parsing).

Gateways are often shared across customers, as a gateway for each and every exchange participant would likely require a massive data-center footprint. As such, gateways must be closely monitored for malicious manipulation. An example of gateway "gaming" is shown in Figure 11. In Figure 11a, client A is connected to two distinct gateways. In 11b, Client A induces extreme load on Gateway 1, causing Client B traffic to slow. In 11c, Gateway 1, not under load, slows all attempts for Client B to cancel resting markets. Client A has an advantage with the self-made fast path.

Matching engine. The matching engine is the core of the exchange, and like its HFT cousins is fairly straightforward. A matching engine, shown in Figure 12, is a simple queue management system, with a queue of bids and a queue of offers. When a customer attempts to execute against a queue, the exchange searches the queue until it reaches the requested size and removes those orders from the queue.

The difficulties arrive in determining who receives notifications first. The aggressing party does not know (for certain) it has traded 10,000 shares until receiving a confirmation. The passive parties do not know they have been executed until they receive a confirmation. Finally, the market as a whole does not know the trade has occurred until the market data is published with the new queue. Problems such as this are becoming increasingly more difficult to solve as we move from milliseconds to microseconds to nanoseconds.

Conclusion

The world of high-frequency trading is rich with problems for computer scientists, but it is fundamental to the new marketplace of automated trading, which is responsible for the majority of transactions in the financial markets today. As HFT moves from milliseconds to microseconds to nanoseconds, the problems becoming increasingly more difficult to solve, and technology must strive to keep up. C

Related articles on queue.acm.org

NUMA (Non-Uniform Memory Access): An Overview
Christoph Lameter
http://queue.acm.org/detail.cfm?id=2513149

FPGA Programming for the Masses
David Bacon, Rodric Rabbah and Sunil Shukla
http://queue.acm.org/detail.cfm?id=2443836

Computing without Processors
Satnam Singh
http://queue.acm.org/detail.cfm?id=2000516

References

1. Arista Networks. 7124FX Application Switch; http://www.aristanetworks.com/en/products/7100series/7124fx/7124fx-development.
2. Arista Networks; 7150 Series 1/10 GbE SFP Ultra-Low Latency Switch; http://www.aristanetworks.com/en/products/7150-series/7150-datasheet.
3. CBS News. 2010. Robot traders of the NYSE. Sixty Minutes Overtime; http://www.cbsnews.com/video/watch/?id=6942497n&tag=related;photovideo.
4. Millar, M. 'Lightning fast' future traders working in nanoseconds. BBC News, 2011; http://www.bbc.co.uk/news/business-15722530.
5. Moyer, L. and Lambert, E. Wall Street's new masters. *Forbes* (Sept. 21, 2009), 40–46; http://www.forbes.com/forbes/2009/0921/revolutionaries-stocks-getco-new-masters-of-wall-street.html.
6. NASDAQ. NASDAQ TotalView-ITCH 4.1, 2013; http://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/nqtv-itch-v4_1.pdf.
7. NASDAQ. OMX Co-Location; http://app.qnasdaqomx.com/e/es.aspx?s=453941583&e=9032&elq=4824c6a202f34d00a5e586d106f64cc8.
8. NASDAQ. OUCH Version 3.1, 2012; http://www.nasdaqtrader.com/content/technicalsupport/specifications/TradingProducts/NQBX_OUCH3.1.pdf.
9. NASDAQ. SoupTCP; http://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/souptcp.pdf.

Jacob Loveless is the CEO of Lucera, a New York-based cloud computing company that services low-latency financial customers. He was a special contractor for the U.S. Department of Defense with a focus on heuristic analysis of things that cannot be discussed.

practice

DOI:10.1145/2507771.2507780

Article development led by acmqueue
queue.acm.org

The challenges faced by competing HFT algorithms.

BY JACOB LOVELESS, SASHA STOIKOV, AND ROLF WAEBER

Online Algorithms in High-Frequency Trading

HIGH-FREQUENCY TRADING (HFT) has emerged as a powerful force in modern financial markets. Only 20 years ago, most of the trading volume occurred in exchanges such as the New York Stock Exchange, where humans dressed in brightly colored outfits would gesticulate and scream their trading intentions. Today, trading occurs mostly in electronic servers in data centers, where computers communicate their trading intentions through network messages. This transition from physical exchanges to electronic platforms has been particularly profitable for HFT firms, which invested heavily in the infrastructure of this new environment.

Although the look of the venue and its participants has dramatically changed, the motivation of all traders, whether electronic or human, remains the same: to buy an asset from a location/trader and to sell it to another location/trader for a higher price. The defining difference between a human trader and an HFT is that the latter can react faster, more frequently, and has very short portfolio holding periods. A typical HFT algorithm operates at the sub-millisecond time scale, where human traders cannot compete, as the blink of a human eye takes approximately 300 milliseconds. As HFT algorithms compete with each other, they face two challenges:

ILLUSTRATION BY SPOOKY POOKA AT DEBUT ART

▸ They receive large amounts of data every microsecond.

▸ They must be able to act extremely fast on the received data, as the profitability of the signals they are observing decays very quickly.

Online algorithms provide a natural class of algorithms suitable for HFT applications. In an online problem, new input variables are revealed sequentially. After each new input the algorithm needs to make a decision—for example, whether or not to submit a trade. This is in stark contrast to an offline problem, which assumes the entire input data is available at the time of the decision making. Many practical optimization problems addressed in computer science and operations research applications are online problems.[1]

Besides solving an online problem, HFT algorithms also need to react extremely fast to market updates. To guarantee a fast reaction time, efficient memory handling becomes a necessity for a live trading algorithm. Keeping a large amount of data in memory will slow down any CPU, so it is important that an algorithm uses only a minimal amount of data and parameters, which can be stored in fast accessible memory such as the L1 cache. In addition, these factors should reflect the current state of the market and must be updated in real time when new data points are observed. In summary, the smaller the number of factors that need to be kept in memory and the simpler the computation required to update each factor, the faster an algorithm is able to react to market updates.

Based on the speed requirement and the online nature of HFT problems, the class of *one-pass algorithms* is especially suitable for HFT applications. These algorithms receive one data point at a time and use it to update a set of factors. After the updating, the data point is discarded and only the updated factors are kept in memory.

In this article we discuss three estimation problems that can arise in HFT algorithms and that can be efficiently solved with a one-pass algorithm. The first is the estimation of a running mean of liquidity; this can be useful to an HFT algorithm in determining the size of an order that is likely to execute successfully on a particular electronic exchange. The second problem is a running violation estimation, which can help quantify the short-term risk of a position. The third problem is a running linear regression, which can be used in trading pairs of related assets.

Online Algorithms in HFT

The one advantage that HFT has over other market participants is reaction speed. HFT firms are able to see every action in the market—that is, the information contained in the limit order book—and react within microseconds. Though some HFT algorithms may base their actions on a source of information outside the market (say, by parsing news reports, measuring temperature, or gauging market sentiment), most base their decisions solely on the processing of messages arriving to the market. By some estimates, there are approximately 215,000 quote updates per second on the New York Stock Exchange.[4] The challenge for HFT algorithms is to process this data in a way that allows them to make decisions, such as when to enter positions or reduce risk. The examples used in this article assume that HFT algorithms can observe every update in the best bid and ask prices, including the best bid and ask sizes. This subset of information contained in the limit order book is often referred to as the Level-I order book information.

The following three examples of online algorithms, each motivated with an application in HFT, are described in detail in this article:

▸ *Online Mean Algorithm.* Illustrated by constructing a factor that predicts the available liquidity, defined as the sum of the sizes at the best bid and the best ask, at a fixed horizon in the future. This quantity may be useful in estimating what order size is likely to execute at the best quotes at a given latency.

▸ *Online Variance Algorithm.* Illustrated by constructing a factor that predicts the realized volatility over a fixed horizon in the future. This quantity may be useful in estimating the short-term risk of holding inventory.

▸ *Online Regression Algorithm.* Illustrated by constructing a factor that predicts the expected PNL (profit and loss) of a long-short position in two related assets. This may be useful in constructing a signal indicating when a long-short position is likely to be profitable.

In all three cases, the algorithm has a single parameter, alpha, which controls the rate at which old information is forgotten. Figure 1 plots the raw liquidity measure (bid size plus ask size) in blue. Red and green represent the online liquidity factor, with alpha=0.9 and alpha=0.99, respectively. Note that as alpha approaches a value of 1, the signal gets smoother and efficiently tracks the trend in the underlying data.

Figure 2 plots the online volatility measure for various values of alpha. Once again, notice that the measure is smoother for the larger alpha. Although

Figure 1. Raw and online liquidity.

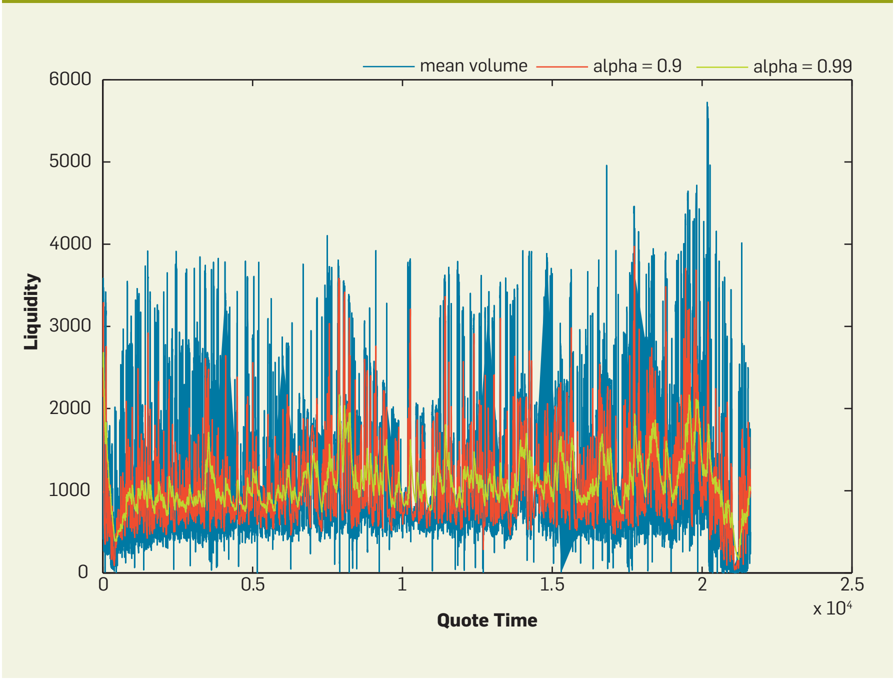
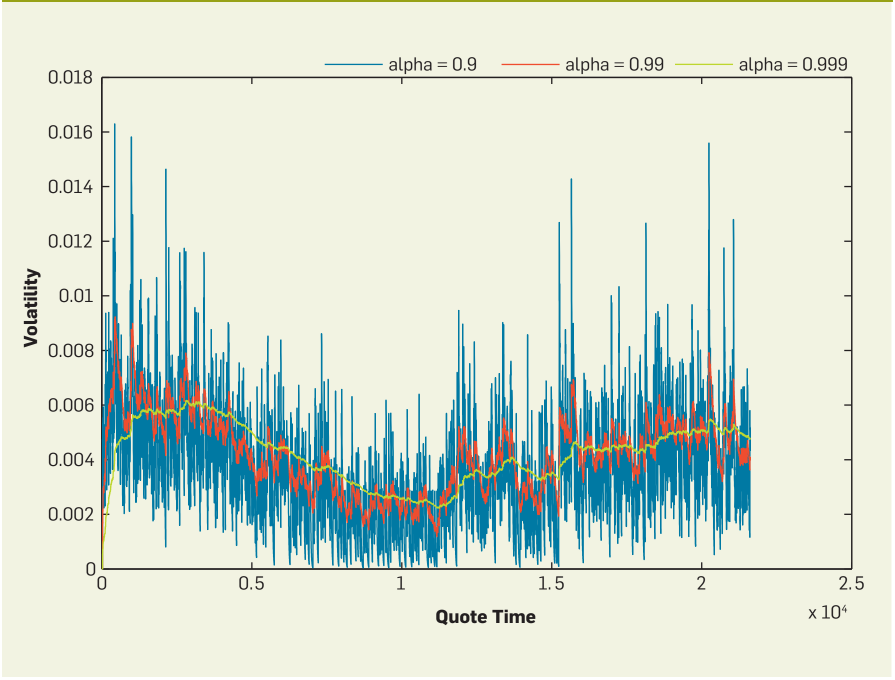


Figure 2. Online volatility measure for various values of alpha.



a larger alpha provides a smoother signal, it also lags further behind the underlying trend as it gives a lot of weight to older data. As discussed later, choosing a value for alpha translates into a trade-off between a smooth signal and a reduced lagging of the trend.

To illustrate the online regression algorithm, we look at the time series of mid prices for SPY and SSO, two highly related ETFs (SSO is the double-leveraged version of SPY). As shown in Figure 3, the relationship between the two assets seems very close to linear over the course of a day. Figure 4 plots the online mean and intercept for two values of alpha.

One-Pass Algorithms

As indicated by its name, a *one-pass algorithm* reads every input variable exactly once and then discards it. This type of algorithm is very efficient in terms of memory handling, as it requires only a minimal amount of data to be stored in memory. Here, we present three important examples of online one-pass algorithms: the exponential moving average, the exponentially weighted variance, and the exponentially weighted regression. Each of these algorithms has been used in a HFT application in the previous section.

First, let's look briefly at the *simple moving average* of a time series. This is an estimate of the mean of a time series over a moving window of a fixed size. In finance, it is often used to detect trends in price, in particular by comparing two simple moving averages: one over a long window and one over a short window. In another application, the average traded volume over the past five minutes can serve as a prediction of the volume traded in the next minute. In contrast to the exponential moving average, the simple moving average cannot be solved with a one-pass algorithm.

Let $(X_t)_t = X_0, X_1, X_2 \ldots$ be the observed sequence of input variables. At any given time t we want to predict the next outcome X_{t+1}. For $M > 0$ and $t \geq M$, the simple moving average with window size M is defined as the average of the last M observations in the time series $(X_t)_t$—that is, $\hat{X}_{t,M} = \frac{1}{M}\sum_{j=0}^{M-1} X_{t-j}$. The moving average can also be computed via the following recursion:

$$\hat{X}_{t,M} = \hat{X}_{t-1,M} - \frac{X_{t-M}}{M} + \frac{X_t}{M}. \qquad (1)$$

While this is an online algorithm, it is not a one-pass algorithm, as it needs to access every input data point exactly twice: once to add it to the moving average and then again to drop it out of the moving average estimate. Such an algorithm is also referred to as a two-pass algorithm and requires keeping an entire array of size M in memory.

Example 1: One-Pass Exponential Weighted Average. In contrast to the regular average $\bar{X}_{t,M} = \frac{1}{t+1}\sum_{j=0}^{t} X_j$, the exponential weighted average assigns an

Figure 3. Online regression algorithm.

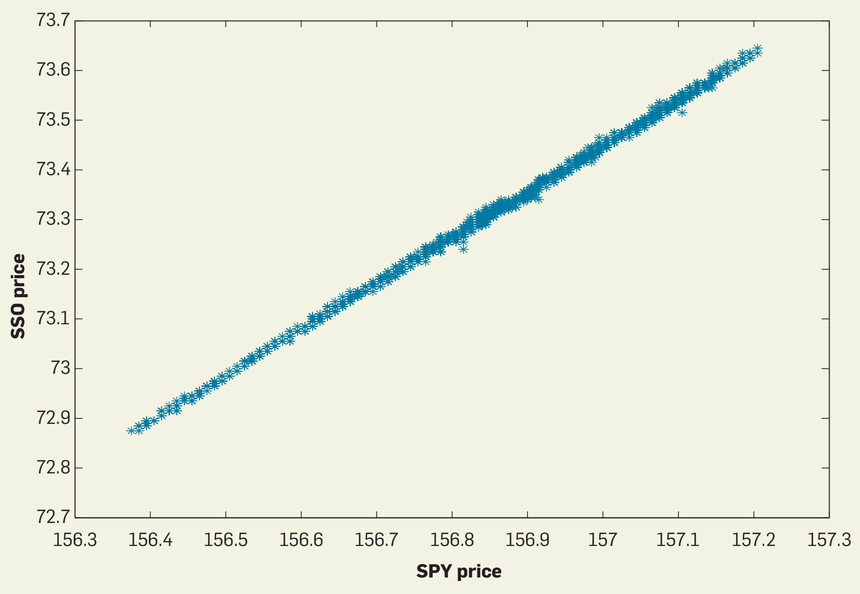


Figure 4. Online mean and intercept for two values of alpha.

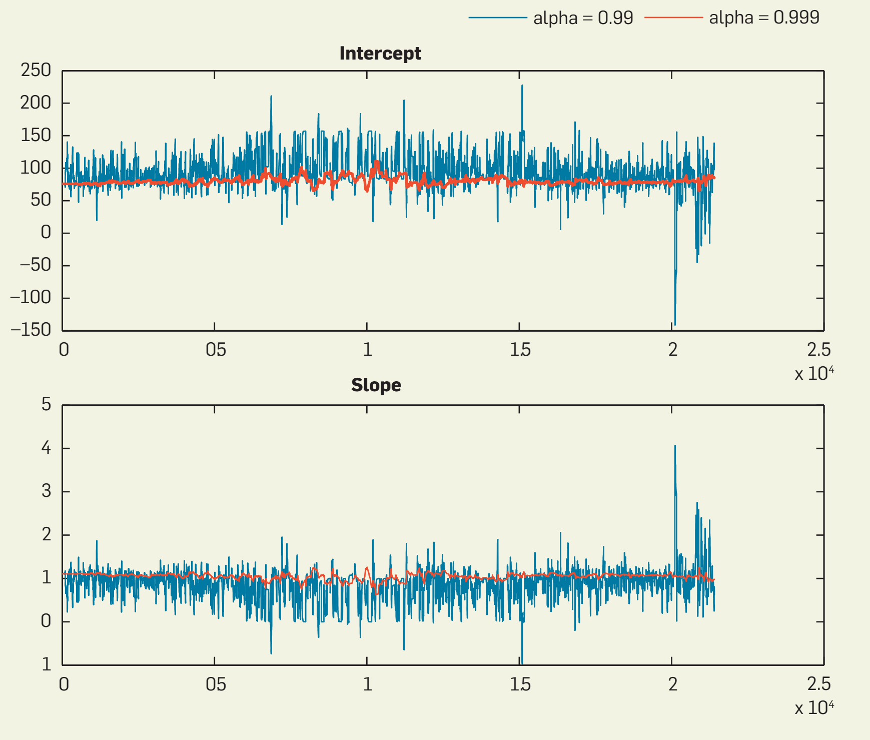


Figure 5. The moving average and the weighting parameter.

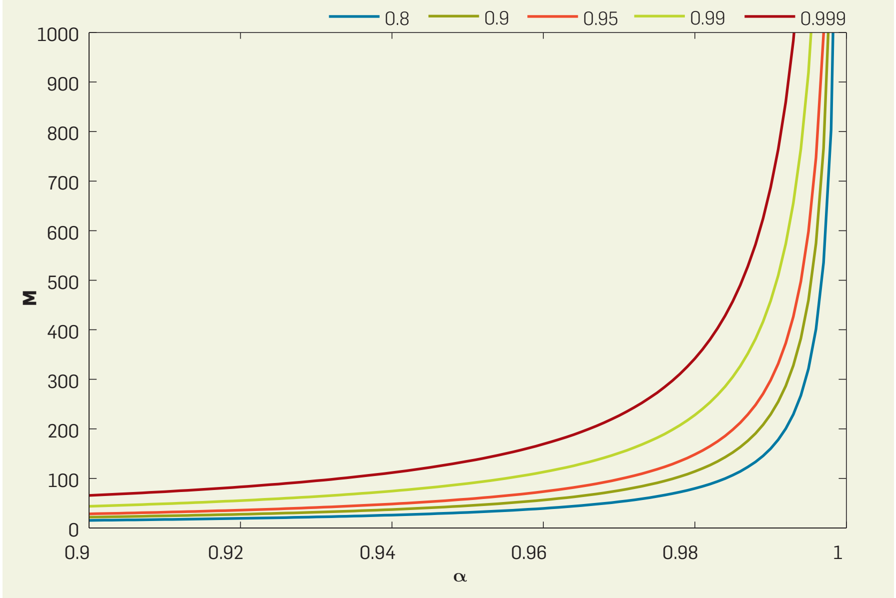


Figure 6. One-pass algorithm for online estimation of exponentially weighted variance.

$$\hat{X}_{0,\alpha} = X_0$$
$$\hat{V}_{0,\alpha} = 1$$
$$\hat{X}_{t,\alpha} = (1-\alpha)X_{t,\alpha} + \alpha\hat{X}_{t-1,\alpha}$$
$$\hat{V}_{t,\alpha} = (1-\alpha)(X_{t,\alpha} - \hat{X}_{t,\alpha})^2 + \alpha\hat{V}_{t-1,\alpha}$$

exponentially decreasing weight to older observations:

$$\hat{X}_{t,\alpha} = (1-\alpha)\sum_{j=0}^{t+1}\alpha^j X_{t-j} + \alpha^t X_0$$

Here α is a weighting parameter chosen by the user and needs to satisfy $0 < \alpha \leq 1$. As this exponential weighted average gives more importance to more recent input compared with older data points, it is often considered to be a good approximation of the simple moving average.

Compared with the simple moving average, the exponential weighted average takes all previous data into account and not just the last M observations. To compare the simple moving average and the exponential weighted average further, Figure 5 shows how many data points receive 80%, 90%, 95%, 99%, and 99.9% of the weight in the estimation as a function of α. For example, if $\alpha = 0.95$, then the last $M = 90$ observed data points contribute to 99% of the estimated value. As a warning, if the time series $(X_t)_t$ has very heavy tails, then the exponential smoothed average might be dominated by an extreme observation, whereas the moving average is less prone to extreme observations as these eventually drop out of the observation window. Frequent restarting of the estimation procedure can solve this long-term memory effect of exponential smoothing.

The reason for favoring the exponential moving average over the simple moving average in HFT is it can be efficiently solved using a one-pass algorithm, initially introduced in Brown.[3]

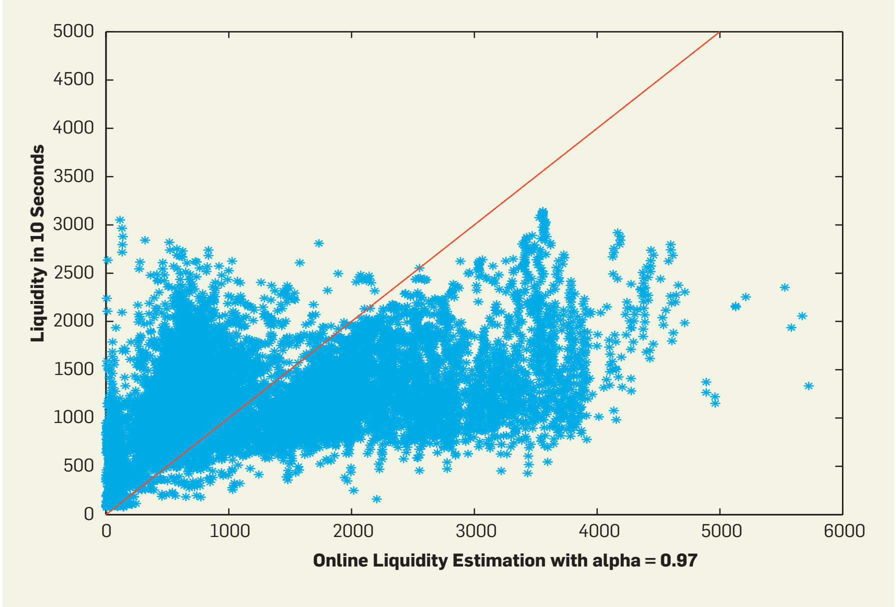
$$\begin{aligned}\hat{X}_{0,\alpha} &= X_0\\ \hat{X}_{t,\alpha} &= (1-\alpha)X_t + \alpha\hat{X}_{t-1,\alpha}\end{aligned} \qquad (2)$$

This formula also provides a simple interpretation of the parameter α as a control of how much weight is given to the most recent observation, compared with all previous observations.

Example 2: One-Pass Exponentially Weighted Variance. The exponential smoothing described in the previous section estimates a moving average of a time series. In finance, the volatility of a time series is often an important factor as well. Broadly speaking, volatility should capture how much a time series fluctuates around its mean. There is no widely accepted definition of volatility for high-frequency financial data. Here, we consider the volatility to be the standard deviation (square root of variance) of a data point in the time series $(X_t)_t$. Similar to the exponentially weighted moving average discussed previously, an online one-pass algorithm can be constructed that estimates the volatility of the time series $(X_t)_t$ with an exponential weighting scheme.

The variance of a random variable is defined as $Var(X) = E[(X - E[X])^2]$. Estimating the exponential weighted variance of the time series requires two estimators: one that estimates the

Figure 7. Scatter plot of factor and response for alpha of 0.97.



mean $E[X]$ and one that estimates the variance as illustrated in Figure 6.

The standard deviation of the next measurement point X_{t+1} is then estimated as $\sqrt{\hat{V}_{t,\alpha}}$. Again, the input parameter $\alpha \in (0,1)$ needs to be chosen by the user and reflects how much weight is assigned to older data points compared with the latest observed data input. Here, we initialized the estimator of the variance with 1, which is a rather arbitrary choice. Another way is to have an initial "burn-in" period for which the time series $(X_t)_t$ is observed and a standard variance estimator of the series over this burn-in time window can be used to initialize the estimator. Of course, a similar method can be used to initialize the estimator of the exponentially weighted average estimator.

Example 3. One-Pass Algorithm for Exponentially Weighted Linear Regression. The last example is an online one-pass algorithm for the exponentially weighted linear regression model. This model is similar to ordinary linear regression, but again gives more importance (according to an exponential weighting) to recent observations than to older observations. As already shown, such regression methods are very useful in HFT strategies to estimate the relation of different assets, which can be, for example, exploited in creating pair trading strategies.

In this model we consider a two-dimensional time series $(X_t,Y_t)_t$ and conjecture that the variables X and Y are related via a linear relation that is corrupted by a noise term ε_t with zero mean. That is,

$$Y_t = \beta_0 + \beta_1 X_t + \varepsilon_t \qquad (3)$$

The variable Y is referred to as the response variable, whereas X is called the explanatory variable. For simplicity let's assume just one explanatory variable here, but the extensions to several explanatory variables are straightforward. In the standard offline approach to linear regression, the parameters β_0 and β_1 are calibrated after all the data points are observed. These data points are collected in a vector $\boldsymbol{Y} = (Y_0, Y_1, \ldots, + Y_t)^T$, and a matrix

$$\boldsymbol{X} = \begin{bmatrix} 1 & X_0 \\ \vdots & \vdots \\ 1 & X_t \end{bmatrix}$$

The column of ones in the matrix X corresponds to the intercept in equation 3. If we further write the parameters and β_0 and β_1 as a vector—that is, $\beta = (\beta_0, \beta_1)^T$ —then the relationship between Y and X can conveniently be written in matrix notation as

$$\boldsymbol{Y} = X\beta + \varepsilon$$

where ε is a vector of stochastic noise terms, and each of these error terms has zero mean.

The most common approach to estimating the parameter β is using ordinary least squares estimation—that is,

Estimating Alpha

How does one decide on the optimal value of alpha, the one parameter of all these online models? Our approach for all three models is to define a response function that we aim to predict, and minimize the squared error between the response r_i and our factor f_i:

$$\min_{\alpha} \sum_i (f_i(\alpha) - r_i)^2$$

This method finds the optimal alpha on a historical time series. Another approach would be to estimate the optimal alpha online as well. This, however, requires more work and goes beyond the scope of this article.

We now provide the details on the online estimators described and estimate the optimal alpha on a given data set.

1. The mean liquidity estimator is defined as

$$f_i^{liq}(\alpha) = f_{i-1}^{liq} + (1-\alpha)\frac{bs_i}{bs_i + as_i}$$

where the index i represents quote time. The response is defined to be the liquidity in 10 seconds:

$$r_i = \frac{bs_i(10)}{bs_i(10) + as_i(10)}$$

where $bs_i(10)$ represents the bid size 10 seconds after the i-th quote. Running an optimization routine over alpha shows that the optimal alpha for the given data is 0.97, displayed in Figure 7 as a scatter plot of the factor and the response.

2. The volatility estimator is defined as

$$m_i(\alpha) = \alpha m_{i-1}(\alpha) + (1-\alpha)(p_i - p_{i-1})$$
$$v_i(\alpha) = \alpha v_{i-1}(\alpha) + (1-\alpha)(p_i - p_{i-1} - m_i(\alpha))^2$$
$$f_i^{vol}(\alpha) = \sqrt{v_i(\alpha)}$$

where the index i represents real time in seconds and the p_i the price of an asset at time i. The response is defined to be the realized volatility over the next minute:

$$r_i = \sqrt{\sum_{j=1}^{60} (p_{i+j} - p_{i+j-1} - \frac{1}{60}(p_{i+60} - p_i))^2}$$

Again, searching over different values of alpha yields an optimal alpha of 0.985 for the given data set. Figure 8 displays a scatter plot of the factor and the response.

3. The pairs trading regression estimator is defined as

$$M_i(\alpha) = \alpha M_{i-1}(\alpha) + (1-\alpha)\begin{pmatrix}1\\ p_i^{sso}\end{pmatrix}(1\ p_i^{sso})$$

$$V_i(\alpha) = \alpha V_{i-1}(\alpha) + (1-\alpha)\begin{pmatrix}1\\ p_i^{sso}\end{pmatrix}\left(p_i^{SPY}\right)$$

$$\beta_i^{reg}(\alpha) = M_i(\alpha)^{-1} V_i(\alpha)$$

$$f_i^{reg}(\alpha) = (1\ p_i^{sso})\beta_i^{reg}(\alpha) - p_i^{SPY}$$

where the index i represents quote time. The factor $f_i^{reg}(\alpha)$ represents the value of SPY relative to SSO—that is, if the quantity is positive, then SPY is relatively cheap and a trade that is long SPY is likely to be profitable.

The response is defined as the PNL over the next minute of a trade that is long one share of SPY and short β shares of SSO:

$$r_i = \left(p_i^{SPY}(60) - \beta_i^{reg}(\alpha)\begin{pmatrix}1\\ p_i^{sso}(60)\end{pmatrix}\right) - \left(p_i^{SPY} - \beta_i^{reg}(\alpha)\begin{pmatrix}1\\ p_i^{sso}\end{pmatrix}\right)$$

where $p_i^{SPY}(60)$ represents the price of SPY 60 seconds after p_i^{SPY}. The response r_i represents the PNL of the following long-short strategy: "Buy 1 share of SPY and sell β shares of SSO at time i, exit the position after 60 seconds."

In the analyzed data set the optimal alpha turns out to be 0.996. Figure 9 is a scatter plot of the factor and the response.

Figure 8. Scatter plot of factor and response for alpha of 0.985.

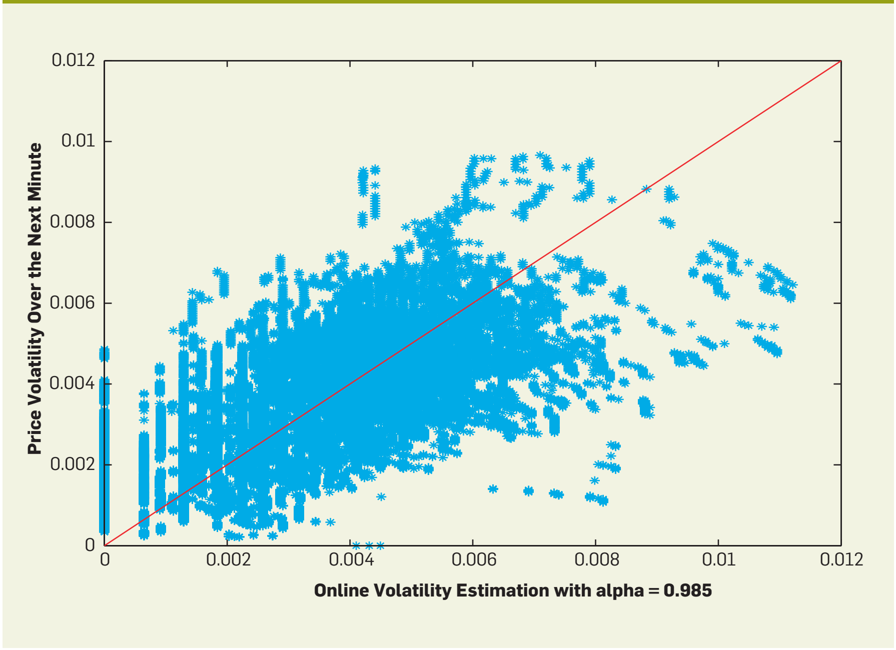
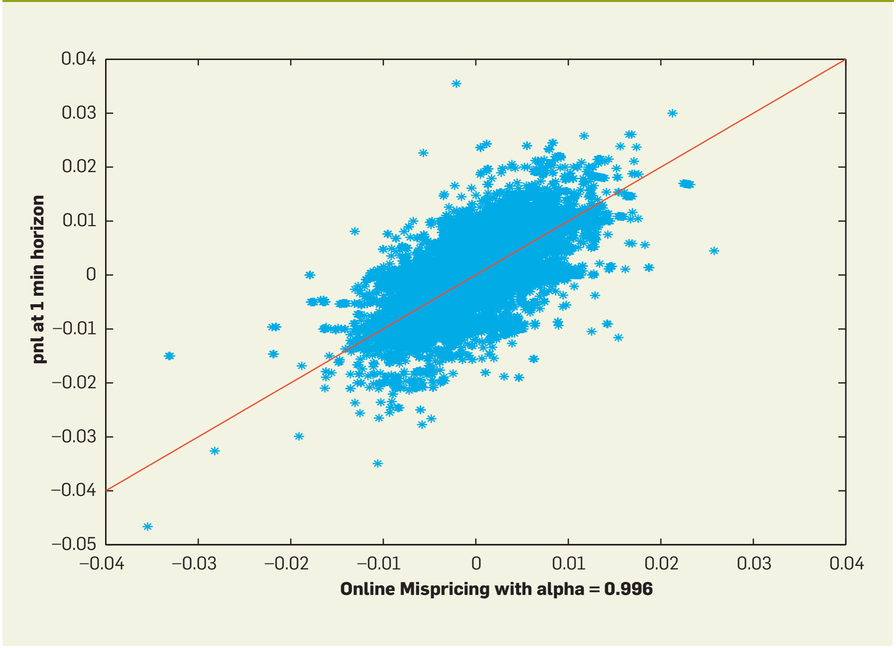


Figure 9. Scatter plot of factor and response for alpha of 0.996.



β is chosen such that it minimizes the sum of squared residuals $\Sigma_{j=0}^{t}(Y_j - (\beta_0 + \beta_1 X_j))^2$. The solution to this minimization problem is $\hat{\beta} = (X^T X)^{-1} X^T Y$.

As in mean and variance estimations, more recent data points should be more important for the estimation of the parameter β. Also a one-pass algorithm of β is required for fast computation.

Next let's consider a recursive method that updates β sequentially and minimizes

$$\sum_{j=0}^{t} \alpha^{(t-j)}(Y_j - (\beta_0 + \beta_1 X_j))^2$$

Again, the parameter α needs to be in the range (0,1) and is chosen by the user. The parameters β_0 and β_1 of the weighted least squares estimation can be computed with an efficient online one-pass algorithm. At each step of the algorithm a 2×2 – matrix $\boldsymbol{M}_t$ and a 2×1 – vector $\boldsymbol{V}_t$ need to be saved in memory and updated with a new data point according to the following recursion:

$$\begin{aligned} \boldsymbol{M}_t &= \alpha \boldsymbol{M}_{t-1} + \boldsymbol{X}_t^T \boldsymbol{X}_t, \\ \boldsymbol{V}_t &= \alpha \boldsymbol{V}_{t-1} + \boldsymbol{X}_t^T \boldsymbol{Y}_t, \end{aligned}$$

As for the mean and variance estimator, the initialization of the recursion can be done using a burn-in period. Finally, after time t, the best estimate of β is $\hat{\beta}_t = M_t^{-1} V_t$. In the literature this method is also called recursive least squares with exponential forgetting.[2]

Conclusion

Online one-pass algorithms are instrumental in HFT, where they receive large amounts of data every microsecond and must be able to act extremely fast on the received data. This article has addressed three problems that HFT algorithms face: the estimation of a running mean of liquidity, which can be useful in determining the size of an order that is likely to execute successfully on a particular electronic exchange; a running volatility estimation, which can help quantify the short-term risk of a position; and a running linear regression, which can be used in trading pairs of related assets. Online one-pass algorithms can help solve each of these problems.

References

1. Albers, S. Online algorithms: A survey. *Mathematical Programming 97*, 1–2 (2003), 3–26.
2. Astrom, A. and Wittenmark, B. *Adaptive Control*, second edition. Addison Wesley, 1994.
3. Brown, R.G. *Exponential Smoothing for Predicting Demand.* Arthur D. Little Inc., 1956, p. 15
4. Clark, C. Improving speed and transparency of market data. *Exchanges*, 2011; https://exchanges.nyx.com/cclark/improving-speed-and-transparency-market-data.

Jacob Loveless is the CEO of Lucera and former head of High Frequency Trading for Cantor Fitzgerald. He has worked for HFT groups and exchanges in nearly every electronic asset. Previously, he was a special contractor for the U.S. Department of Defense and CTO and a founder of Data Scientific.

Sasha Stoikov is a senior research associate at Cornell Financial Engineering Manhattan (CFEM) and a former VP in the HFT group at Cantor Fitzgerald. He has worked as a consultant at the Galleon Group and Morgan Stanley and was an instructor at the Courant Institute of NYU and at Columbia's IEOR department.

Rolf Waeber is a Quantitative Research Associate at Lucera and previously served as a Quantitative Researcher at Cantor Fitzgerald's HFT Group. He participated in studies on liquidity risk adjustments within the Basel II/III regulation frameworks at the Deutsche Bundesbank.

DOI:10.1145/2507771.2507781

Article development led by acmqueue
queue.acm.org

A close look at round-trip time measurements with the Transmission Control Protocol.

BY STEPHEN D. STROWES

Passively Measuring TCP Round-Trip Times

MEASURING AND MONITORING network round-trip time (RTT) is important for multiple reasons: it allows network operators and end users to understand their network performance and help optimize their environment, and it helps businesses understand the responsiveness of their services to sections of their user base.

Measuring network RTT is also important for Transmission Control Protocol (TCP) stacks to help optimize bandwidth usage. TCP stacks on end hosts optimize for high performance by passively measuring network RTTs using widely deployed TCP timestamp options carried in TCP headers. This information, if utilized, carries some distinct operational advantages for services and applications: hosts do not need to launch out-of-band Internet Control Message Protocol (ICMP) echo requests (pings), nor do they need to embed timing information into application traffic. Instead, hosts can passively measure RTT representative of full-path network latency experienced by TCP traffic.

Understanding network delay is key to understanding some important aspects of network performance. The time taken to traverse the network between two hosts affects how responsive services are, and it affects the effective bandwidth available to end hosts. Measuring this information passively on servers can help provide a fine-grained indication of service responsiveness from the customer's perspective, and it simultaneously offers a network-distance metric for customers that is more useful than coarse-grained and often inaccurate IP-based geolocation.

Measuring the RTT to many hosts or customers is nontrivial. One solution is active probing, in the form of ICMP echo requests and responses (that is, ping), but this incurs additional network load. In some cases ICMP traffic is deprioritized, dropped completely, or routed over a different path to TCP traffic. If none of this is true, there is still the possibility that the RTT is being measured to a Network Address Translator (NAT) and not the end host exchanging data.

Another possible solution for measuring network RTT is to measure the application-layer responsiveness. This, however, implies an application-specific, ad hoc measurement performed by embedding IDs or timestamps into the TCP bytestream, which may give a misleading, inflated measurement for network RTT, depending on network conditions (more on this later).

Neither of these solutions is totally satisfactory because neither is guaranteed to measure the network RTT that affects application traffic. The timestamp information carried in TCP headers, however, offers another solution: it is effectively a network-layer RTT measurement that passes through most middleboxes such as NATs and firewalls and measures the full-path RTT between both hosts on a connection. This information provides the only noncustom RTT estimation solution available to end hosts. Tools such as tcptrace can calculate RTT using this state, but any software that can inspect packet headers (usually accomplished via libpcap) or that can interrogate the local system for such kernel state can passively gather network RTTs for all active connections.

The key differences between these measurements and how differing network conditions affect them are not obvious. The purpose of this article is to discuss and demonstrate the passive RTT measurements possible using TCP traffic.

Background

TCP offers a reliable bytestream to the applications that use it; applications send arbitrary amounts of data, and the TCP layer sends this as a series of segments with sequence numbers and payload lengths indicating the chunk of the bytestream each segment represents. To achieve an ordered byte stream, TCP retransmits segments if they go missing between the source and destination (or, if an acknowledgment for a received segment goes missing between the destination and the source). To improve performance in all network conditions, the TCP stack measures RTT between it and the other host on every connection to allow it to optimize its retransmission timeout (RTO) appropriately and optimize the time taken to recover from a loss event.

The original TCP specification contained no mandatory, dedicated RTT calculation mechanism. Instead, TCP stacks attempted to calculate RTTs by observing the time at which a sequence number was sent and correlating that with a corresponding acknowledgment. Calculation of RTTs using this mechanism in the presence of packet loss, however, makes accurate measurement impossible.[13] TCP timestamps were defined to permit this calculation independently at both ends of a connection while data is being exchanged between the two hosts. TCP timestamps offer a mechanism for calculating RTT that is independent of sequence numbers and acknowledgments.

The algorithm for calculating RTT from a TCP flow between two hosts, documented in RFC 1323,[3] is commonly used by both end hosts on a connection to refine the RTO to improve the performance of TCP in the presence of loss. The mechanism is enabled by default in modern operating systems and is rarely blocked by firewalls, and thus appears in most TCP flows; the TCP Timestamp option is known to be widely deployed in the wild.[5]

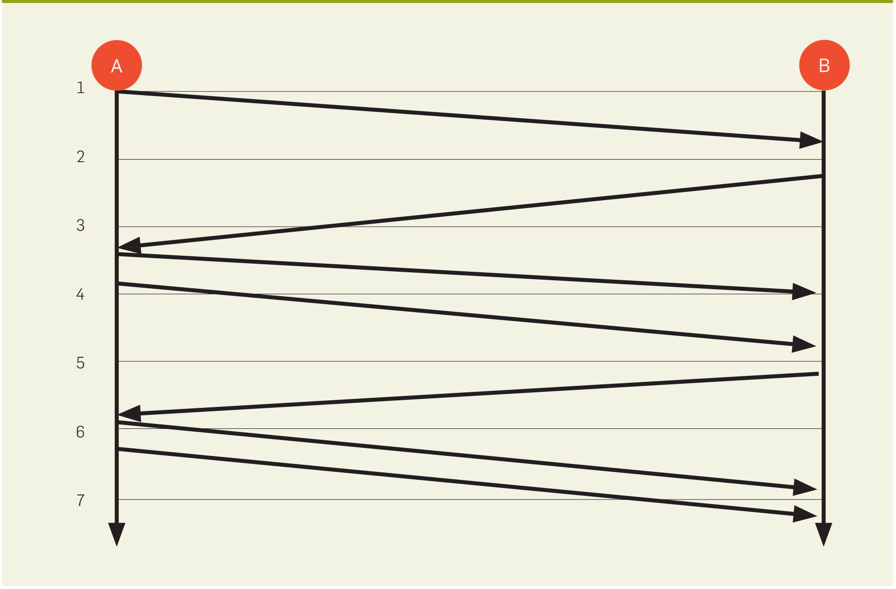
RTT is calculated continuously for each connection for as long as data is exchanged on those connections. TCP calculates RTT for packets exchanged on a per-connection basis and computes the exponential moving average of these measurements, referred to as the *smoothed* RTT (SRTT). The TCP stack also maintains the variance in the measured RTT, the RTTVAR. The SRTT that the TCP stack calculates for a connection determines the RTO value. Given variables `G`, which is the system clock granularity, and `K`, which is set to 4,[8] the RTO is calculated as follows:

```
RTO = SRTT + max(G,K * RTTVAR)
```

The RTO is used to optimize the time the TCP stack waits, having transmitted a TCP segment, for a corresponding acknowledgment prior to retrying the send operation. Accurate measurements of RTT between two hosts allow TCP to tune its RTO accurately for each active connection.

Understanding the state contained within the TCP headers carried in most TCP traffic can help application designers or network operators understand the network RTTs experienced by application traffic. Many applications with real-time constraints use TCP to transport their traffic, which is acceptable within certain bounds.[1] It is useful to understand

Figure 1. Simplified demonstration of TCP timestamp exchange.



how the bytestream semantic can affect real-time performance.

TCP timestamps are optional fields in the TCP header, so although they are extremely useful and carried in most traffic, they are not strictly required for TCP to function. The values are held in two four-byte header fields: Timestamp Value (TSval) and Timestamp Echo Reply (TSecr). Both hosts involved in the connection emit TSval timestamps to the other host whenever a TCP segment is transmitted, and they await the corresponding TSecr in return. The time difference measured between first emitting a TSval and receiving it in a TSecr is the TCP stack's best guess as to RTT. *Timestamp* here is an arbitrary value that increments at the granularity of the local system clock; it is not a timestamp that can be interpreted independently, such as number of seconds since the epoch.

By way of example, in Figure 1, time progresses from top to bottom, and the horizontal lines indicate real-time incrementing (for example, in milliseconds). Two hosts, A and B, have an open connection and are exchanging packets. In reality the two hosts have differing clocks, but for simplicity assume they are perfectly synchronized.

The example operates as follows:

Host A emits a TCP segment that contains the timestamp options

```
TSval = 1, TSecr = 0
```

TSecr is set to 0 because no TSval from B has been observed at A; this usually indicates A is opening a connection to B. Host B receives this timestamp at time 1; at time 2, host B emits a TCP segment back to A, which contains the values

$\texttt{TSval = 2, TSecr = TSval}^{A}\texttt{ = 1}$

These are received at host A at time 3. Given this echoed value and the current time, host A knows that the RTT in this instance is approximately 2ms.

Subsequently, the next two segments that A emits both carry the values:

$\texttt{TSval = 3, TSecr = TSval}^{B}\texttt{ = 2}$

The first of these is received at host B at time 4, so host B can also calculate an RTT of 2ms. Given the two echoes of the same timestamp received, the minimum duration is assumed to be closest to actual network delay; if network delay changes, future exchanges will measure this. Continuously sending values to the other host and noting the minimum time until the echo reply containing that value is received allows each end host to determine the RTT between it and the other host on a connection.

To improve performance in all network conditions, the TCP stack measures RTT between it and the other host on every connection to allow it to optimize its retransmission timeout (RTO) appropriately.

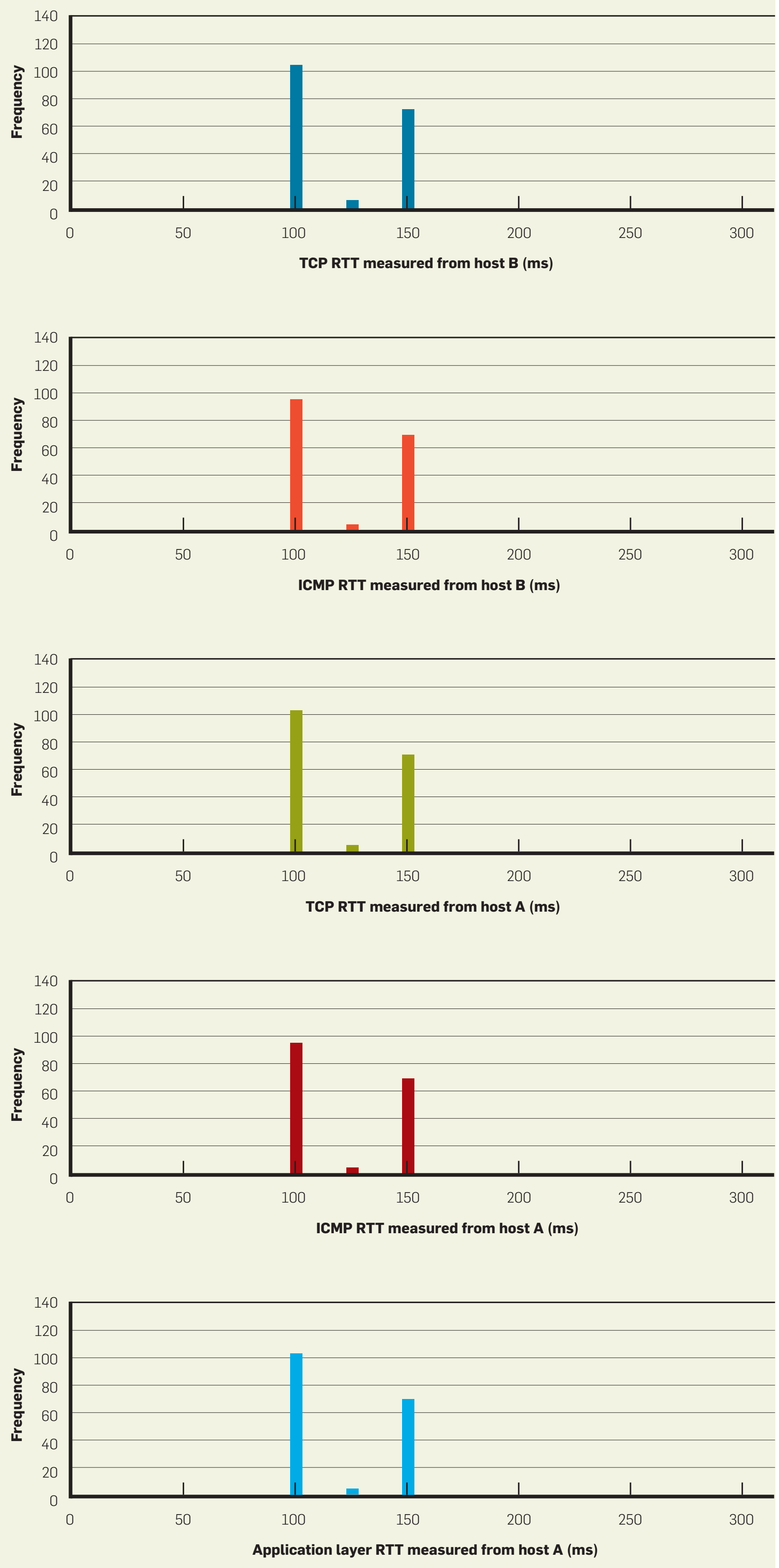
The caveat is that for a TSval to be considered useful, the TCP segment must be carrying data from the application. TCP segments can legitimately carry a zero-byte payload, most commonly when acknowledging a data segment, or when TCP keepalives are enabled. By requiring the only valid TSvals come from TCP segments carrying data, the algorithm is less likely to measure breaks in the communication between the hosts, where data exchange pauses for a time, then restarts using the last received TSval as the TSecr. This also implies that on a TCP flow in which data is exchanged exclusively in one direction, only one of the hosts will be able to calculate the RTT. Usually, however, there is some application layer chatter in both directions.

Finally, RTT calculation can be performed on any host that is forwarding traffic, not just end hosts, so full-path RTTs on all connections within a network can be calculated from its gateway host, for example. All that is necessary to compute accurate RTTs is that both directions of a connection pass through the monitoring host. Whether this is the case often relies on the network architecture, but it is known that paths on the Internet are normally not symmetric.[2] Running this algorithm on a gateway node for a network through which all traffic passes, however, allows for the RTT calculation to take place passively on all connections from just one location.

Demonstrating RTT Measurements

The network is a shared resource, and multiple factors can affect TCP RTT calculation. This section broadly covers some of these factors and demonstrates where the TCP RTT calculation differs from the RTT perceived by applications. The aim is to demonstrate parity with ICMP's RTT estimations, assuming all else is equal, and how

Figure 2. Histograms indicating measured RTTs for all tests.



packet loss and excessive buffering affect these measures relative to perceived latency at the application layer.

To demonstrate the responsiveness of RTT measurements, traffic flows were simulated in a virtualized environment. The environment is simple: two Linux hosts were configured on different subnets, with a third Linux host with packet forwarding enabled, configured with two network interfaces, one for each subnet.

This forwarding host is used to vary the network characteristics observed between the two end hosts, using the traffic control (tc) tool. Network characteristics are not modified on the end hosts, so their TCP stacks are not directly aware of the configuration for each experiment. For each experiment, an egress delay of 50ms is set on each interface on the forwarding host, resulting in an RTT of 100ms between the two end hosts. Each experiment runs for 180 seconds. The maximum data rate is set to 10Mbps.

On these end hosts, the following components are running:

- Ping is running on both hosts, so each host is sending ICMP echo requests to the other once per second. This measures the ICMP RTT value to establish a "ground-truth" RTT between the hosts.
- A simple client/server pair of programs is running, where the client sends a local timestamp over a TCP connection once every second to the server, and the server echoes the timestamp back to the client; the client calculates the difference whenever it reads the response out of the byte stream. The client application runs two threads: one for sending and one for receiving. This measures the RTT perceived by the application layer.
- Also running on both end hosts is a packet capture (pcap) reader that observes the TCP timestamp values carried in the TCP headers for the traffic generated by the client/server program from which it calculates the RTT, outputting the latest RTT value once every second. The value exported for these experiments is the RTT rather than the SRTT, since the goal here is to examine the actual RTT and not an approximation. This calculates the RTT passively from TCP timestamps. No other traffic is exchanged between

hosts, except during the demonstration of bufferbloat.

The following examples demonstrate:

1. The ability to monitor changing RTT accurately by modifying network latency.
2. The impact of packet loss.
3. The impact of oversized buffers (commonly referred to as bufferbloat).

Nagle's algorithm. Before describing these experiments in detail, we should take a look at Nagle's algorithm,[6] which is enabled by default in many TCP stacks. Its purpose is to reduce the number of small, header-heavy datagrams transferred by the network. It operates by delaying the transmission of new data if the amount of data available to send is less than the MSS (maximum segment size), which is the longest segment permissible given the maximum transmission unit on the path, and if there is previously sent data still awaiting acknowledgment.

Nagle's algorithm can cause unnecessary delays for time-critical applications running over TCP. Thus, because the assumption is that such applications will run over TCP in the experiments presented here, Nagle's algorithm is disabled. This is achieved in the client and server by setting the TCP_NODELAY socket option on all sockets in use.

Experiment 1: Changing Network Conditions. When computing RTTs, it is critical the measurements accurately reflect current conditions. The purpose of this experiment is simply to demonstrate the responsiveness of our metrics to conditions that change in a predictable manner. In this experiment the base RTT (100ms) is initially set, and then an additional latency (50ms) is alternately added and deducted from that base RTT by incrementing the delay on both interfaces at the forwarding host by 25ms. No loss ratio is specified on the path, and no additional traffic is sent between the two hosts.

Note that since TCP's RTT calculation is wholly passive, it does not observe variation in RTT if no data is being exchanged. In the presence of traffic, however, it's beneficial that the RTT measurement update quickly. The results of this experiment are shown in Figure 2. The measurements taken at all layers indicate a bimodal distribution, which is precisely what we should expect without other network conditions affecting traffic. The three forms of measurements taken are all effectively equivalent, with the mean RTT measured during the experiments varying by no more than 1%.

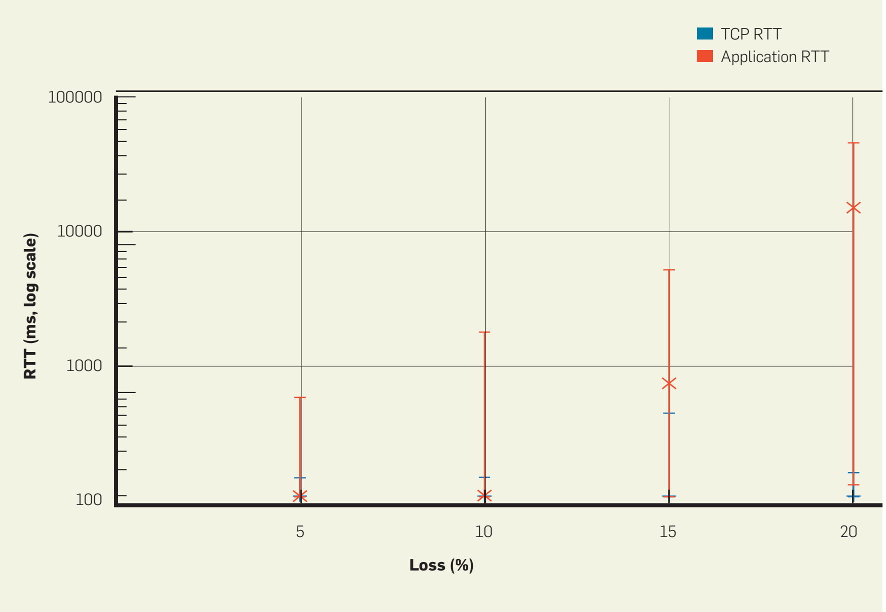
Experiment 2: Packet Loss. Packet loss on a network affects reliability, responsiveness, and throughput. It can be caused by many factors, including noisy links corrupting data, faulty forwarding hardware, or transient glitches during routing reconfiguration. Assuming the network infrastructure is not faulty and routing is stable, loss is often caused by network congestion when converging data flows cause a bottleneck, forcing buffers to overflow in forwarding hardware and, therefore, packets to be dropped. Loss can happen on either the forward or the reverse path of a TCP connection, the only indication to the TCP stack being the absence of a received ACK.

TCP offers applications an ordered bytestream. Thus, when loss occurs and a segment has to be retransmitted, segments that have already arrived but that appear later in the bytestream must await delivery of the missing segment so the bytestream can be reassembled in order. Known as head-of-line blocking, this can be detrimental to the performance of applications running over TCP, especially if latency is high. Selective acknowledgments, if enabled, allow a host to indicate precisely which subset of segments went missing on the forward path and thus which subset to retransmit. This helps improve the number of segments "in flight" when loss has occurred.

In this experiment, packet loss was enabled on the forwarding host at loss rates of 5%, 10%, 15%, and 20%, the purpose being to demonstrate that TCP segments are still exchanged and RTTs estimated by TCP are more tolerant to the loss than the RTTs measured by the application. The results of this experiment are shown in Figure 3. The points represent median values, with 5th and 95th percentiles shown.

In these tests, a 5% packet loss was capable of introducing a half-second delay for the application, even though the median value is close to the real RTT of 100ms; the mean measured application layer RTT with 5% loss is 196.4ms, 92.4ms higher than the measured mean for TCP RTT. The measured means rise quickly: 400.3ms for 10% loss, 1.2s for 15% loss, and 17.7s for 20% loss. The median values shown in Figure 3 for application-layer RTT follow a similar pattern, and in this example manifest in median application-layer RTTs measured at around 12 seconds with 20% packet loss. The TCP RTT, however, is always close to the true 100ms distance; although delayed packet exchanges can inflate this measure, the largest mean deviation observed in these tests between TCP RTT and ICMP RTT was a

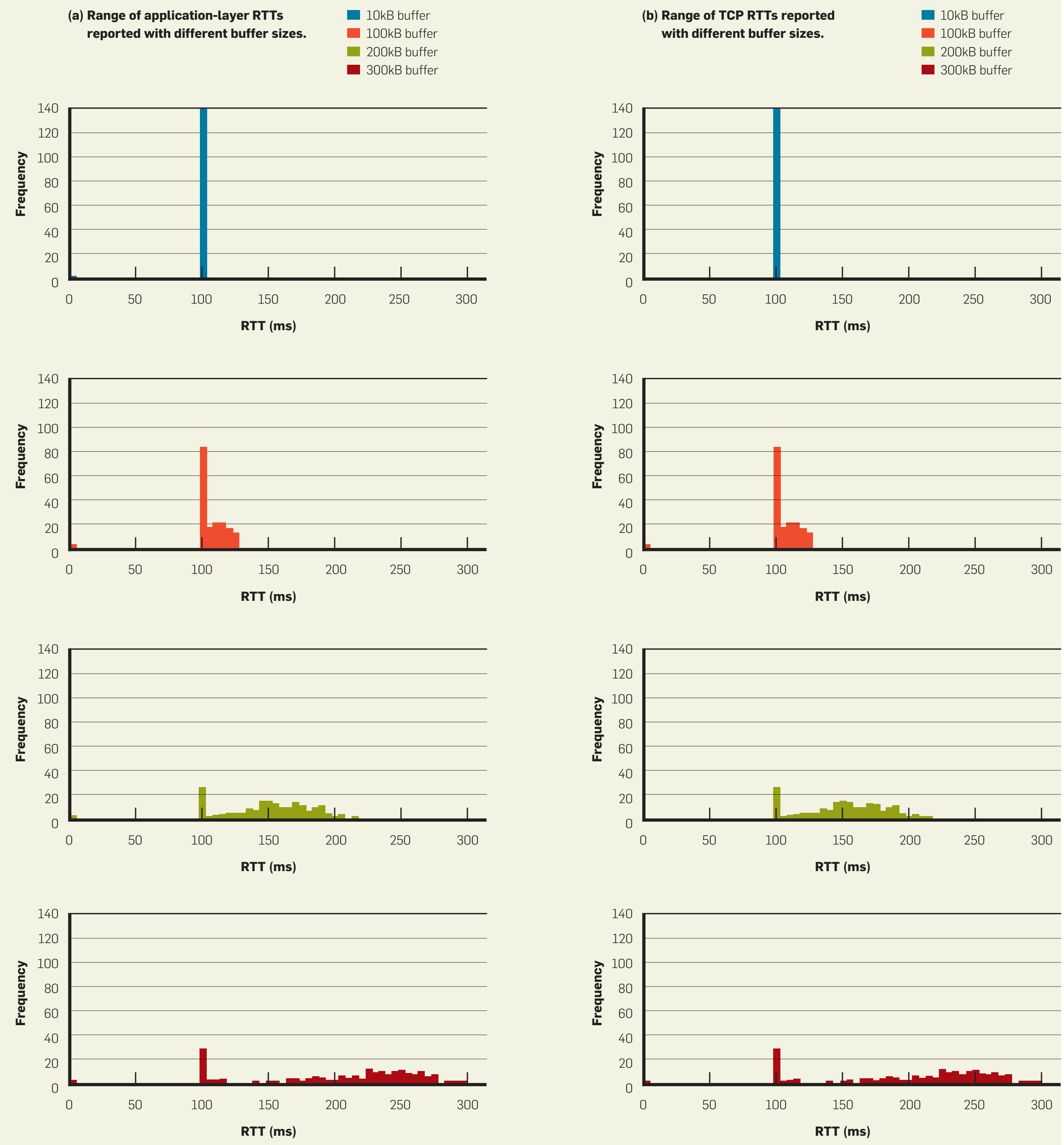
Figure 3. RTTs measured in the presence of varying packet loss rates.



57.7ms increase in measured RTT at the TCP layer. The effect of packet loss can be devastating to the responsiveness of TCP applications, but it is clear that passively measuring network level RTTs is still feasible, and distinct from the perceived latency experienced by applications that can be introduced by TCP's in-order delivery semantics.

Experiment 3: Bufferbloat. Misunderstandings around the relationship between loss prevention and network performance have led to excessive buffering being introduced to forwarding and routing hardware as a loss-avoidance strategy. Often (but not exclusively) this affects commodity customer premises equipment (CPE), and thus directly affects end users. However, excessive buffering works against TCP's loss-detection algorithm by increasing delay and thus delaying the time taken for a TCP stack to identify loss and back-off; that is, the additional delay introduced by large buffers can disrupt TCP's congestion-control mechanism.

Figure 4. RTT histograms representing the difference in RTTs.



Bufferbloat is a well-known phenomenon,[7] where the deepest buffer on a network path between two hosts is eventually filled by TCP. Ostensibly, system designers increase buffer size to reduce loss, but deeper buffers increase the actual time taken for packets to traverse a path, increasing the RTT and delaying the time it takes for TCP to determine when a loss event has occurred. Loss is the driver for TCP's congestion-control algorithm, so increasing buffer size is actually counterintuitive.

To demonstrate bufferbloat in this experiment, tc queue sizes were systematically increased from 10kB, to 100kB, then 200kB, then finally 300kB on the forwarding host, and netcat was used to create a high-bandwidth flow between each of the end hosts prior to starting the client/server application. The intention of the high-bandwidth flow was to fill the longer queues on the forwarding host, demonstrating that the draining time affects application responsiveness.

The results of the experiment are shown in figures 4 and 5. Figure 4 shows the dispersion of RTT measurements as the buffer sizes were increased. Focusing on the 300kB test in Figure 5, we see very similar RTT measures are evident from both hosts in the ICMP measurements, at the TCP layer, and in the application layer; mean and median values for all layers in these experiments were all within 2ms of each other. All RTT measures are inflated by the same amount because the excessive buffer size effectively increases the network-layer path length. Given that the test application only emits a handful of packets once per second, the sawtooth pattern is indicative of the netcat data filling a queue then TCP waiting for the queue to drain prior to sending more of netcat's data, forming a bursty pattern. These filled queues adversely affects the delivery of all other traffic and our test application suffers RTTs, which vary from 100ms to about 250ms as a result.

The bufferbloat problem is being actively worked on. Mechanisms such as Selective Acknowledgments (SACK), Duplicate SACK (DSACK), and Explicit Congestion Notification (ECN), when enabled, all help alleviate bufferbloat. Additionally, active queue management strategies such as Codel have been accepted into mainline Linux kernels.

In summary, it is clear that to minimize delays caused by head-of-line blocking in TCP, packet loss must be kept to a minimum. Given that we must expect packet loss as a primary driver of TCP's congestion control algorithm, we must also be careful to minimize network buffering, and avoid the delays incurred by bufferbloat. The latter requirement in particular is useful to keep in mind when provisioning networks for time-critical data that must be delivered reliably.

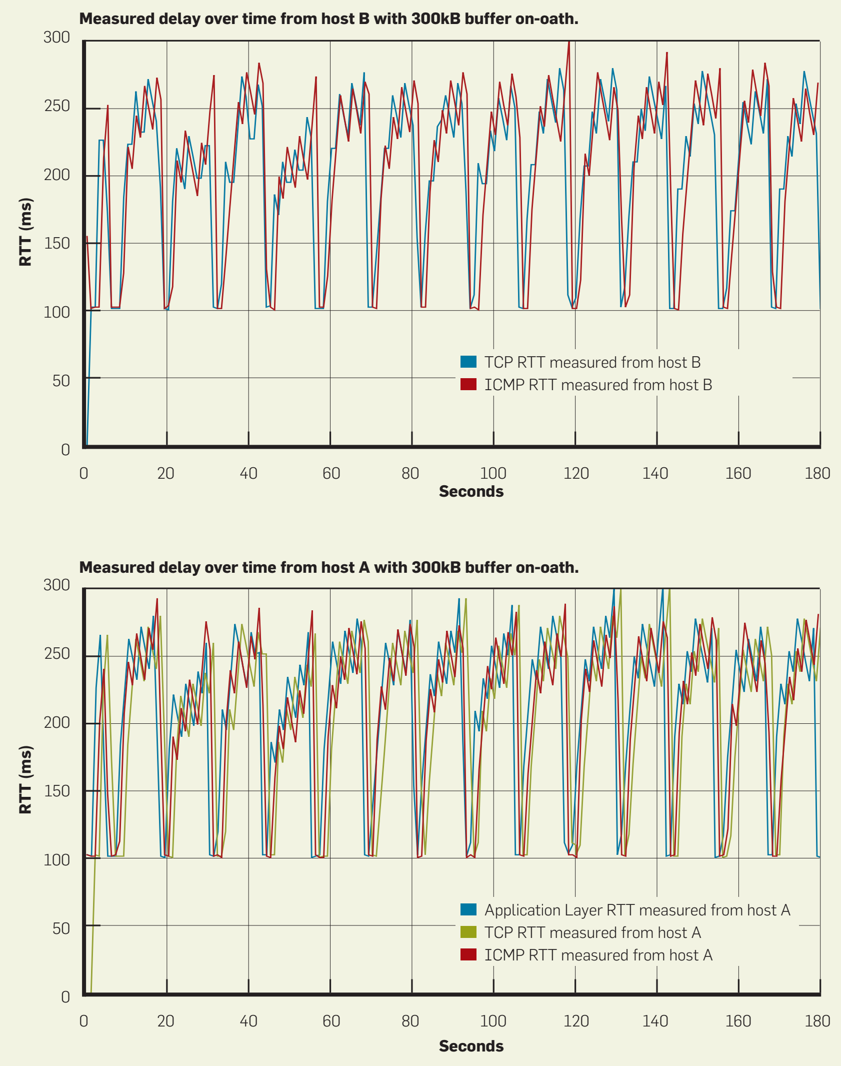
Related Work

The key issue when using TCP for time-sensitive applications is that TCP offers a reliable bytestream. This requirement is distinct from other key aspects of TCP, such as congestion control and flow control. TCP is not suitable for all applications, however. Eli Brosh et al. discuss in more detail the behavior of TCP in the presence of delay and certain acceptability bounds for application performance.[1]

UDP[9] is the most commonly used transport protocol after TCP; it's a datagram-oriented protocol with no congestion control, flow control, or message-ordering mechanisms. It effectively augments the IP layer with UDP-layer port numbers. Without the message-ordering constraint, it is not affected by the head-of-line blocking problem that can affect TCP connections.

UDP alone is not suitable for many applications, however, because reli-

Figure 5. Indication of measured RTT values in the presence of excessive buffering.



ability is often a requirement, and congestion control is important to permit fair sharing of network resources. Many applications choose to layer protocols on top of UDP, such as RTP Real Time Protocol (RTP) in tandem with Real Time Control Protocol (RTCP),[10] primarily intended for carrying time-sensitive real-time traffic able to handle small amounts of loss. These protocols suit applications such as VoIP that do not require 100% reliability and find delay incurred by head-of-line blocking detrimental. RTCP permits coarse-grained congestion control and allows real-time applications to modify their usage by choosing different quality settings for the live stream, but congestion control is not built in per se.

DCCP[4] is a message-oriented, best-effort transport-layer protocol that does not enforce strict ordering on data delivery, does not handle datagram retransmission, but does perform congestion control to conserve network resources. DCCP is useful for a similar set of applications as RTP and RTCP, but the addition of congestion control without potential datagram duplication is important, permitting RTP to run over DCCP with fewer concerns for network resource consumption.

SCTP[11] is also a message-oriented transport, where each message is delivered to the application in-order. Strict message ordering, however, is optional, and so the transport can be more responsive for application traffic. SCTP also caters for partial reliability.[12]

Note that bufferbloat is endemic, and other transport protocols are affected in the same way as TCP, but relaxing strict ordering constraints at the transport layer is one approach to improving performance by removing the additional response time incurred when the stream is blocked waiting for missing data. Active queue management (AQM) techniques[7] are being deployed in new Linux kernels to help further alleviate bufferbloat without modification to applications.

Conclusion

TCP is the most commonly used transport-layer protocol today, and it meets the requirements that many applications desire: it offers a reliable bytestream and handles concerns of retransmissions and congestion avoidance. TCP's semantics can mean that there is a large discrepancy between the RTT measured at the transport layer and the RTT measured by the application reading the bytestream. Thus, TCP is not always the most applicable transport for time-critical applications, but the TCP RTT measurement mechanism that is enabled in most TCP stacks today achieves measurements very close to the ICMP "ground truth" and performs substantially better than a similar echo-based protocol embedded within the TCP bytestream.

TCP is the most commonly used transport-layer protocol today, and it meets the requirements that many applications desire: it offers a reliable bytestream and handles concerns of retransmissions and congestion avoidance.

Related articles on queue.acm.org

Bufferbloat: Dark Buffers in the Internet
Jim Gettys and Kathleen Nichols
http://queue.acm.org/detail.cfm?id=2071893

TCP Offload to the Rescue
Andy Currid
http://queue.acm.org/detail.cfm?id=1005069

References

1. Brosh, E., Baset, S., Misra, V., Rubenstein, D. and Schulzrinne, H. The delay-friendliness of TCP for realtime traffic. *IEEE/ACM Transactions on Networking 18*, 5 (2010), 478-491.
2. He, Y., Faloutsos, M., Krishnamurthy, S. and Huffaker, B. On routing asymmetry in the Internet. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)* 2005.
3. Jacobson, V., Braden, R. and Borman, D. TCP extensions for high performance. RFC 1323.
4. Kohler, E., Handley, M. and Floyd, S. Datagram Congestion Control Protocol. RFC 4340 (2006).
5. Kühlewind, M., Neuner, S. and Trammell, B. 2013. On the state of ECN and TCP options on the Internet. *Passive and Active Measurement.* M. Roughan and R. Chang, eds. *Lecture Notes in Computer Science 7799* (2013). Springer Berlin Heidelberg, 135–144.
6. Nagle, J. Congestion control in IP/TCP internetworks. RFC 896 (1984).
7. Nichols, K. and Jacobson, V. Controlling queue delay. *Queue 10*, 5 (2012); http://queue.acm.org/detail.cfm?id=2209336.
8. Paxson, V., Allman, M., Chu, J. and Sargent, M. Computing TCP's retransmission timer. RFC 6298 (2011).
9. Postel. J. User Datagram Protocol. RFC 768 (1980).
10. Schulzrinne, H., Casner, S. Frederick, R. and Jacobson, V. RTP: A transport protocol for real-time applications. RFC 3550 (2003).
11. Stewart, R. Stream Control Transmission Protocol. RFC 4960 (2007).
12. Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., Conrad, P. Stream Control Transmission Protocol (SCTP) Partial Reliability Extension. RFC 3758 (2004).
13. Zhang, L. Why TCP timers don't work well. In *Proceedings of 1986 SIGCOMM.*

Stephen Strowes is a senior engineer at Boundary Inc., where he works on network measurement metrics and tools. He has previously studied scalable inter-domain routing protocols, NAT traversal protocols, and peer-to-peer protocols for real-time content.

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:10.1145/2500891

Improve online public discourse by connecting opinions across blogs, editorials, and social media.

BY FLORIS BEX, JOHN LAWRENCE, MARK SNAITH, AND CHRIS REED

Implementing the Argument Web

ARGUMENT AND DEBATE are cornerstones of civilized society and intellectual life. As online activity usurps many traditional forms of communication, we would hope to see these processes alive and well on the Web as well. But we do not. Too many mechanisms for online interaction hamper and discourage debate, facilitating poor-quality argument and fuzzy thinking. Needed are new tools, systems, and standards engineered into the heart of the Web to encourage debate, facilitate good argument, and promote a new online critical literacy. This is the Argument Web vision, involving a Web platform combining linked argument data with software tools that make online debate intuitive for its participants, including mediators, students, academics, broadcasters, and bloggers.

New opinions are constantly being presented on websites, blogs, news sites, and discussion forums, challenged and evaluated by a diverse worldwide user group. An important problem is the semantic structure of argumentative viewpoints; for example, whether one person agrees with another or whether a particular statement conflicts with a claim is not captured. A further problem is there is no representation of how arguments across the Web relate to one another and contribute to the overall picture. Despite the numerous vocal communities on the Web, they remain relatively isolated because opinions are not connected.

Needed is an infrastructure allowing for interconnected arguments to be posted anywhere on the Web through a comprehensive underlying ontology of argument. This is the Argument Web,[13,14] a URI-addressable structure of linked argument data making it possible to follow a line of argument (on a particular topic or by a particular person) across disparate forums, comments, editorials, and multimedia resources. A number of bespoke tools have been developed as part of the Argument Web implementation. Various annotation and analysis tools have been developed for academics and trained discourse analysts. Moreover, while argument analysis may be a specialized skill, most people conduct an argument, give reasons, conclude, and give grounds for disagreement every day; it is this intuitive skill the Argument Web aims to support explicitly. Familiar interfaces (such as

» key insights

- **The Web's focus on popularity (such as reflected through "like" buttons and number of followers) instead of rationality (such as through "agree" or "disagree" buttons and number of people subscribing to an opinion) discourages online discourse.**
- **The Web's numerous vocal communities (such as bloggers, Reddit, and Twitter) are rather isolated, with opinions expressed on one website not directly connected to opinions expressed on other websites.**
- **The Argument Web includes new tools, systems, and standards for linking argument data, allowing opinions to be connected across the Web through semantically meaningful links.**

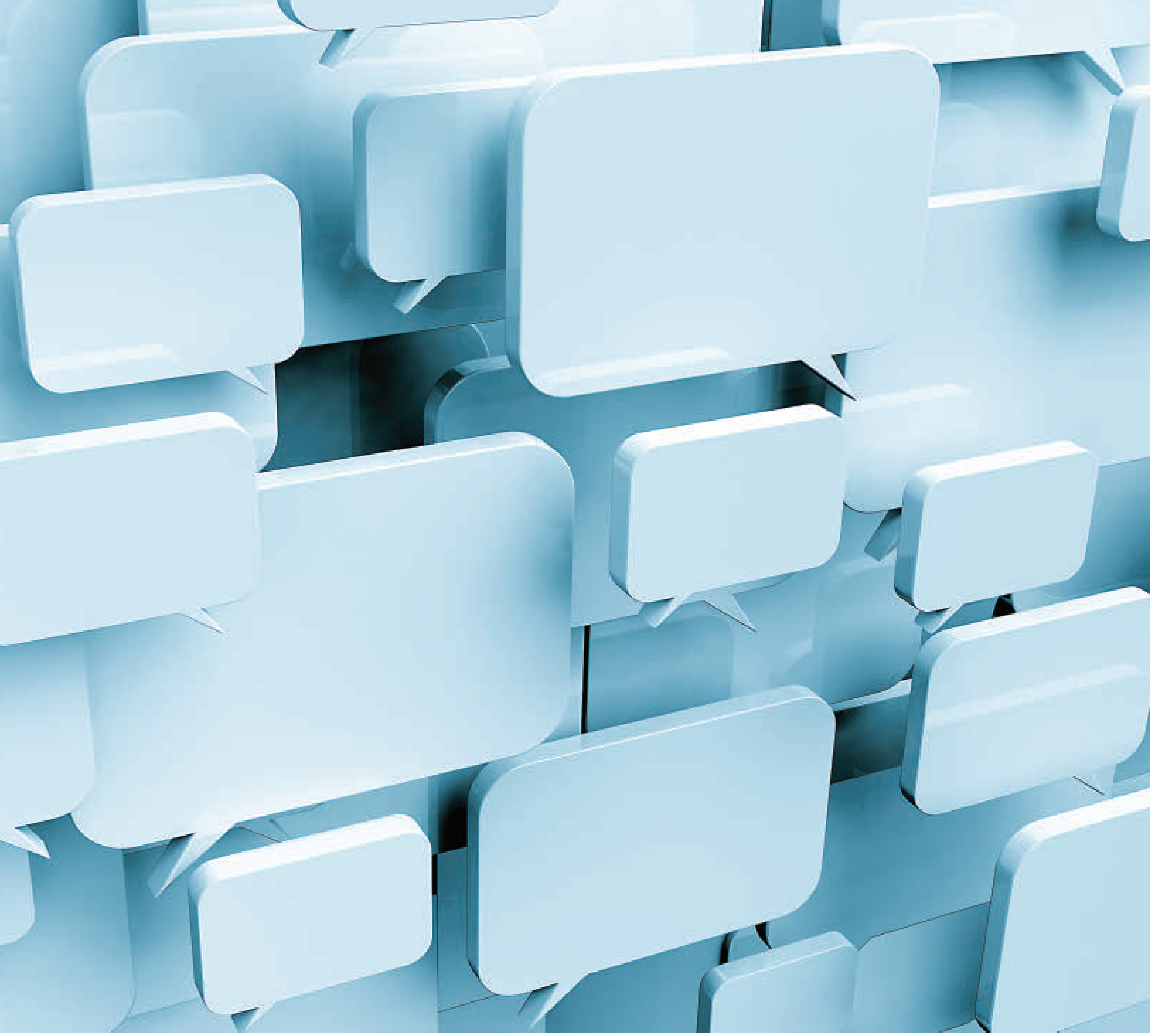


ILLUSTRATION BY CARSTEN REISINGER

blogging and instant messaging) have been adapted, allowing users to navigate opinions and express agreement or disagreement.

In 2007, some of the basic ideas behind the Argument Web were first expressed by Rahwan et al.[14] This article and corresponding webpage (http://www.argumentinterchange.org) explore the first full prototype, discussing a mature version of the argument ontology, implementation of the underlying linked infrastructure, and tools allowing interaction with the Argument Web.

Connecting Opinions

One of the main functions of the Web is enabling people to share, comment on, and argue on a variety of topics, from the newest video game to who should be the next president. Popular blogs can have large numbers of followers who comment on the blog and on each other; discussion forums (such as Reddit) allowing people to share and discuss topics and incorporate mechanisms for rating contributions and users; and Twitter and Facebook allowing users to quickly share and comment on their friends' opinions. While online interaction is facilitated and promoted, online critical discussion usually is not. Comments are rated almost solely according to popularity, not according to whether they present a valid rational argument; a funny picture of President Barack Obama is more likely to end up on the front page of a website than a cogent argument for why one should vote for or against him. It may be possible to "like" a comment, but one cannot "agree" or "disagree" with a comment. Moreover, the only type of relation between two statements is usually "reply" (statement 1 replies to statement 2) and more specific argumentative relations (such as statement 1 supports statement 2 and statement 1 opposes statement 2) are not available.

The need for better online argument and debate is recognized by websites supporting online critical thinking and structured debate. Some offer databases of structured argument for users to explore. Debatabase[a] (formerly Debatepedia) offers numerous high-quality debates about a range of topics, including those for and against a particular topic. Archelogos[b] concentrates on arguments from ancient philosophy (such as Aristotle and Plato), making explicit their argumentative structure and logical interconnections. Other websites are more interactive, allowing users to construct arguments in a structured way. Truthmapping[c] is a more interactive discussion board that introduces slightly more structure to a debate, requiring new statements either support or oppose an existing statement. In Debategraph,[d] debates are visualized as radial trees, with a central topic node surrounded by related statement nodes. Users click nodes to expand elements of a debate and connect new nodes to the tree via relations (such as respond, support, and oppose). A graphical presentation like the one in Debategraph can give a quick overview of complex debates,[e] helping users make sense[11] of a large amount of information; other examples of websites offering visual representations of arguments are Argunet[f] and Cohere.

These Web technologies all have an explicit semantic structure that connects the statements and arguments in a debate, allowing for far better navigation and analysis of a debate; for example, we can render the structure as a graph or as an outline report, count the number of arguments for and against a claim and thus evaluate argument strength, analyze the graph and identify circular reasoning or gaps in chains of arguments, and discover inconsistencies among arguments and agreements among disputants.

The main limitation of these websites is there is little or no integration between them and they do not fully connect to the Web as a whole. Links to sources on the Web can be incorporated into arguments, and debates can be shared by providing links to separate discussion threads or debate graphs. However, these links contain no explicit semantics. Hence, each site provides a silo of structured and semantically rich argumentative content, but these silos are not connected to one another or to the rest of the Web, at least not through semantic links.

To illustrate the added value of connected arguments across the Web, consider the issue of Bashar al-Assad's Syria and the morality of potential Western intervention in the summer of 2012. Say you ask, "Should we invade Syria?" Ideally, you would find a number of arguments, along with links to their sources:

Video. A YouTube video of a press conference in which the U.K. Prime

a http://idebate.org/debatabase
b http://archelogos.com
c http://truthmapping.com
d http://debategraph.org

e For examples of argument and debate analyses, see the "Can Computers Think" argument at http://www.macrovu.com/CCTGeneralInfo.html and http://debategraph.org/can_computers_think and the U.K. nuclear weapons debate at http://www.florisbex.com/NuclearArgMap.pdf
f http://argunet.org

Figure 1. Example of linked argument data.

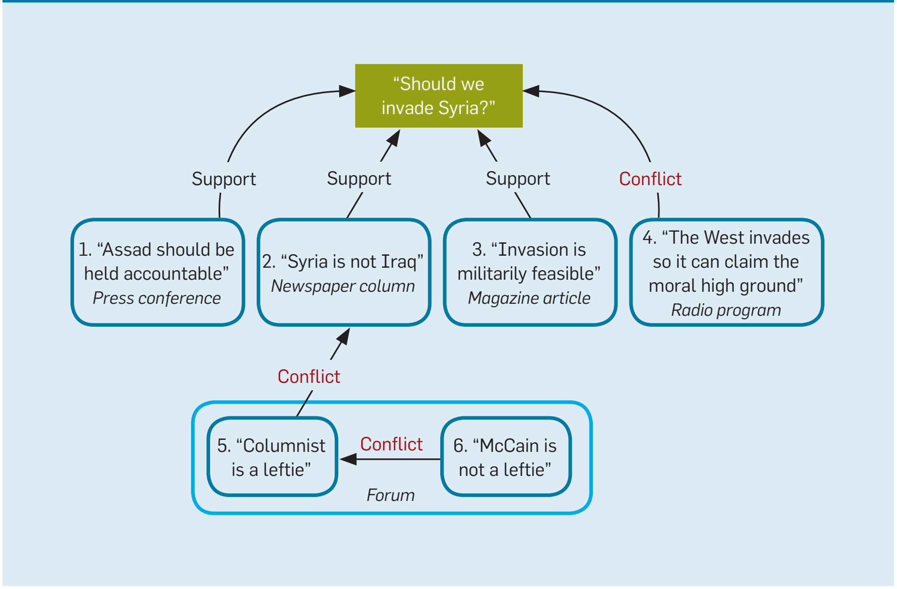
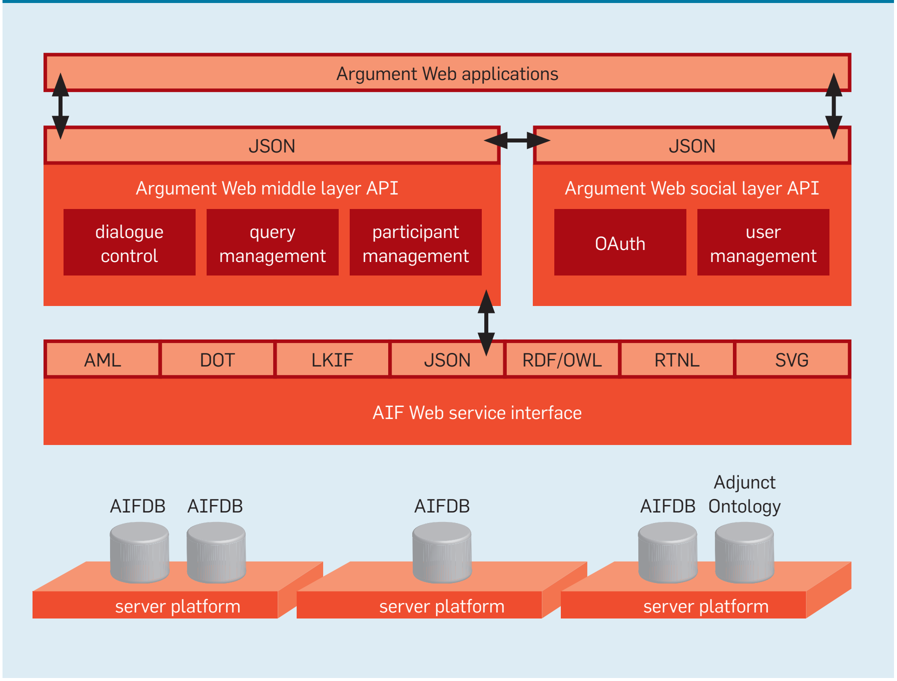

"Should we invade Syria?"
Support
Support
Support
Conflict
1. "Assad should be held accountable"
Press conference
2. "Syria is not Iraq"
Newspaper column
3. "Invasion is militarily feasible"
Magazine article
4. "The West invades so it can claim the moral high ground"
Radio program
Conflict
5. "Columnist is a leftie"
Conflict
6. "McCain is not a leftie"
Forum

Figure 2. The Argument Web architecture.



Minister says Assad should be held accountable for war crimes;[g]

Newspaper. A column in the online edition of *The Guardian* claiming Syria is different from Iraq;[h]

Magazine. An article in *Foreign Policy* exploring the military feasibility of attacking Assad's Syria;[i] and

Radio. A point about the morality of intervening put forward in the BBC Moral Maze program;[j]

(You focus on the newspaper column, which links to other arguments posted on the Web agreeing or disagreeing with the fact that Syria is different from Iraq.)

Forum. One counterargument posted on a forum[k] says the writer of the column is a "leftie" with no idea what is the difference between Syria and Iraq or what real war means; and

On the same forum, another user disagrees with this argument, saying "Arizona Senator John McCain is for intervention and is no leftie."

Figure 1 outlines the small Web of linked arguments, along with support and conflict links. Note the statements and arguments can be on different webpages, but a counterargument can also be given on the same page, as in the forum. In order to realize linked arguments across the Web we need to conform to the demands of linked data,[4] in that each piece of data, or the nodes and links between them, is addressable through a unique URI that stands in well-defined relationships to other URI-addressable data. Only two projects—Cohere[l,7] and the Argument Web—allow for linked argument data.

Cohere aims to link ideas on the Web. Individual online statements can be referred to through a URL, supporting a range of semantic relationships between components (such as "explains" or any other relation the user wants to define). While this breadth is helpful for Cohere, it makes it difficult to build tools with specific applications; for example, argumentation requires a fixed set of argumentative relationships that can then support computational processing (such as visualization, navigation, and evaluation) not easily supported if the relationships being captured are dynamic.

It is possible to use the Argument Web to explore, say, mathematical aspects of arguments phrased in natural language from various sources on the Web.

Infrastructure of the Argument Web

The Argument Web aims to create a Web infrastructure that allows for storage and automatic retrieval and analysis of linked argument data.[4] It is based on a common ontology for argument called the Argument Interchange Format[6] (AIF) that ties together natural linguistic models of argument (such as models that see argumentation as a language activity[17]) with abstract mathematical models of argument.[2,9] It is possible to use the Argument Web to explore, say, mathematical aspects of arguments phrased in natural language from various sources on the Web.

AIF Ontology. At its core, the AIF ontology distinguishes between information (such as propositions and sentences, or the nodes in Figure 1) and general patterns of reasoning that, applied to specific information, provide the individual relations between information (the links in Figure 1). Links can be classified according to the scheme they fulfill.[13] The AIF scheme taxonomy is based on argumentation schemes[18] that are generally accepted for scholarly investigation of argument; for example, the counterargument in Figure 1 saying the columnist is a "leftie" is a typical ad hominem argument,[17] an argument against a person. Here, it is not what the columnist says—Syria is different from Iraq—that is being countered but the credibility of the speaker; that is, people who are left-of-center politically should not be taken seriously when commenting on invading other countries.

In addition to argument structures, or static structures representing information and the support and attack links between them, as in Figure 1, the AIF ontology is also able to capture the argument processes, or the dynamic discussions in which people put forward and challenge claims and reasons. In recent work we showed the AIF ontology can be used as a

g http://youtube.com/watch?v=OBPcv7qSIi4
h http://guardian.co.uk/commentisfree/2012/feb/10/syria-not-iraq-wrong-intervene
i http://foreignpolicy.com/articles/2012/01/10/the_syrian_invasion
j http://bbc.co.uk/programmes/b01kkp5q#synopsis
k http://amazon.com/forum/politics?_encoding=UTF8&cdForum=Fx1S3QSZRUL93V8&cdThread=TxIAO1K6FZP0SI
l http://cohere.open.ac.uk

general framework for capturing dialogue protocols.[3] Such protocols provide rules that determine which types of responses can be given to which types of statements or questions; for example, a challenge statement like "Why is invading Syria militarily feasible?" can force other parties to give reasons for their claim, as in, "We should invade Syria because it is militarily feasible."

Note there is no restriction on the content of an argument; for example, argument contents may themselves be expressed in a common formal ontology, meaning they yield to additional computational processing. On the other hand, this increases the complexity of the ontology, meaning common understanding of the ontology is reduced, and potential users are largely restricted to scholars and experts. Hence, the AIF core ontology is kept as basic as possible, and the Argument Web does not demand explicit semantic characterization of argument content.

Semantic Web. The Argument Web essentially represents a large-scale deployment of Semantic Web[1,13] technology, taking a pragmatic approach to issues of infrastructure by specifying the ontology not only in the formal languages of the Semantic Web (such as the Web Ontology Language, or OWL-DL) but as an instance of a relational database schema adumbrated by a set of Web services acting as an RDBMS for interacting with the data as if it were an RDF triple store. This approach builds on highly scalable, mature, robust, commercially accepted database systems while still conforming to the main principles and demands of linked data.[4]

Argument Web resources are distributed across multiple databases, instances of what are called AIFdb (a database solution for the Argument Web[10]) in reference to the underlying AIF ontology (see Figure 2). Each AIFDB instance provides interfaces through a variety of formats for programmatic access, Semantic Web processing, visualization, and compatibility with existing argumentation tools.[11] Core argumentation functionality is then provided by the Argument Web Middle Layer in three areas:

Query management. Allowing developers to query, say, what argument(s) supports a given claim or whether a given position conflicts with any other(s);

Participant management. Allowing developers to query, say, which users have put forward a given argument; and

Dialogue control. Allowing developers to execute dialogue protocols and, in that context, query whose turn it is in a given dialogue and what legal moves are available to the participants. The Argument Web social layer provides developers a straightforward library through which to construct applications that interact with the Argument Web.

Even rhetorically and linguistically sophisticated maneuvers (such as ad hominem argumentation) can be captured in the Argument Web.

Because the Argument Web, like the Web itself, is designed for public use, the data being produced is noisy, thus presenting a challenge when trying to build computational systems able to reason over Argument Web resources. One approach to dealing with noisy data is engineering oriented: Design the reasoning systems in the full expectation the data over which they work is noisy, and therefore results are either only as good as the data from which they are drawn or else drawn from subsets preprocessed to reduce noise.

The Argument Web's core concept—argumentation—also provides another approach to dealing with noisy data. Take claim identity. In any large distributed knowledge base updated by many different individuals, a major challenge is how to identify when two items in the knowledge base are the same. However, the claim that, say, "the domineering Western aggressors invading Syria" is the same as "Western liberating forces intervening in Syria" is precisely the sort of claim that is liable to be the source of a dispute. Thus, what seems to be an engineering challenge—identifying duplicates—becomes a subject of discussion on the Argument Web.

Argumentation Tools and Interfaces

A number of tools have been developed as part of the Argument Web implementation. Some (such as for argument visualization and analysis) are geared mostly toward educational uses. However, in order to open up the Argument Web to a broader audience, familiar interfaces (such as blogging and instant messaging) can be adapted to support argumentation.[8] Moreover, the solid theoretical and infrastructural basis of the Argument Web allows us to experiment with new technologies for discourse and argument analysis, or Argument Web app library,[m] discussed in the following sections.

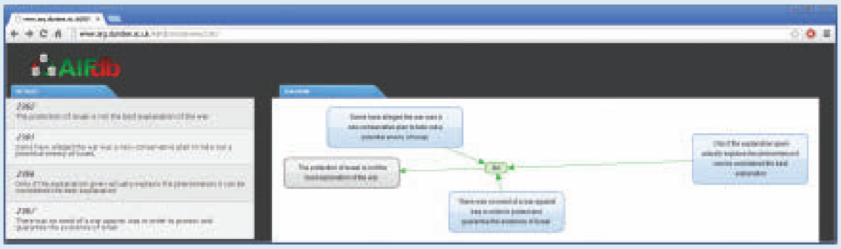
Argument visualization and analysis. As noted, Web technologies for argumentation focus in part on visualizing arguments as graphs or diagrams easily navigated by users. The idea of online argument visualization stems from offline argument-diagramming tools used in, say, courses on critical thinking and for teaching complex legal argumentation.[12] The Argument Web implementation recognizes existing approaches to argument visualization by defining and implementing import-export functions[2,8] for existing argument visualization tools.[n]

Using a visual interface to AIFDB (see Figure 3a) lets users view and navigate the Argument Web; like Debategraph, nodes can be clicked to expand new parts of a debate. New arguments can be constructed using the Online Visualization of Argument, or OVA, allowing direct analysis of Web content; text from a website can be selected and input directly to an argument graph that is then saved to the Argument Web.

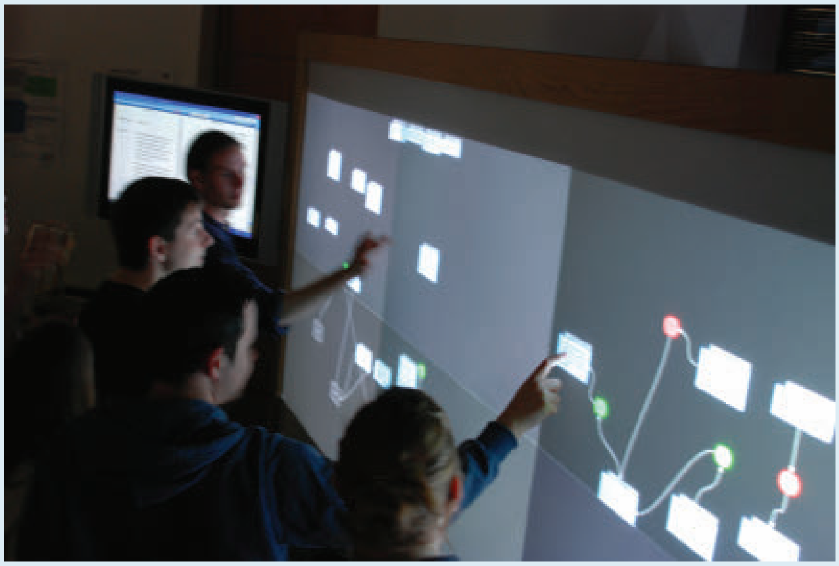
For the University of Dundee's partnership with the BBC, we wanted to analyze a discussion in real time. The BBC's Radio 4 program "The Moral Maze" broadcasts 45 minutes of debate on a moral issue each week. The idea is if an analysis is done in real time and saved directly to the Argument Web, listeners would be able to interact with the discussion as it happens. However, a trained analyst can take weeks to analyze even one hour of debate, so analysis must be performed in parallel. We therefore designed and built a collaborative workspace—the AnalysisWall, a touchscreen measuring 11 feet by 7 feet running bespoke analysis software[14] (see Figure 3b). As the program was broadcast live, stenographers provided a text feed, segmentation analysts broke the text into its component parts, and eight analysts collaborated

m http://www.argumentinterchange.org/library
n http://argumentinterchange.org/developers

Figure 3. Argument visualization and analysis.

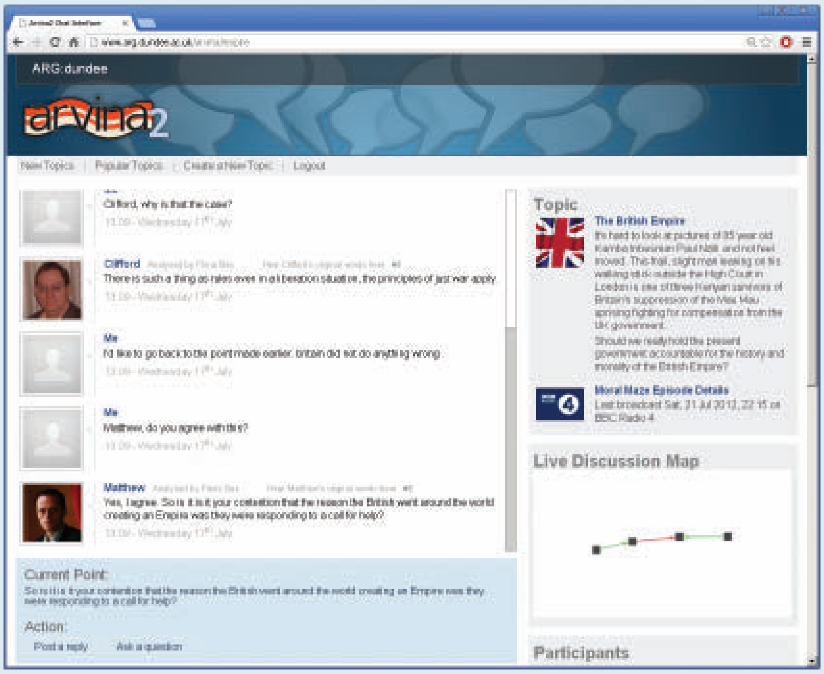


(a)



(b)

Figure 4. The Arvina 2 debate interface.



ARG:dundee

arvina2

New Topics | Popular Topics | Create a New Topic | Logout

Clifford, why is that the case?
13:09 - Wednesday 17th July

Clifford
There is such a thing as rules even in a liberation situation, the principles of just war apply
13:09 - Wednesday 17th July

Me
I'd like to go back to the point made earlier. britain did not do anything wrong
13:09 - Wednesday 17th July

Me
Matthew, do you agree with this?
13:09 - Wednesday 17th July

Matthew
Yes, I agree. So is it is it your contention that the reason the British went around the world creating an Empire was they were responding to a call for help?
13:09 - Wednesday 17th July

Current Point:
So is it is it your contention that the reason the British went around the world creating an Empire was they were responding to a call for help?

Action:
Post a reply   Ask a question

Topic

The British Empire
It's hard to look at pictures of 85 year old Kamba tribesman Paul Nzili and not feel moved. This frail, slight man leaning on his walking stick outside the High Court in London is one of three Kenyan survivors of Britain's suppression of the Mau Mau uprising fighting for compensation from the UK government.
Should we really hold the present government accountable for the history and morality of the British Empire?

Moral Maze Episode Details
Last broadcast Sat, 21 Jul 2012, 22:15 on BBC Radio 4

Live Discussion Map

Participants

to tease apart the structure of what is being said, directly inserting it into the Argument Web.

Direct discussion and mixed-initiative argumentation. Direct discussion between two or more people on the Web takes place not just via email and instant messaging but also on forums and message boards. However, these technologies offer only the most basic of structural tools, and the inferential structure of an argument in a discussion is easily lost. The Web-based discussion software Arvina[10] (see Figure 4) allows participants to debate a range of topics in real time in a way that is structured but at the same time is unobtrusive. Users can ask questions (such as "Do you agree?" and "Why do you think this is the case?") of other participants in the discussion, as well as express their own agreement or disagreement with a particular point and provide supporting reasons for their views. Moreover, Arvina can use dialogue protocols to structure the discussion between participants.

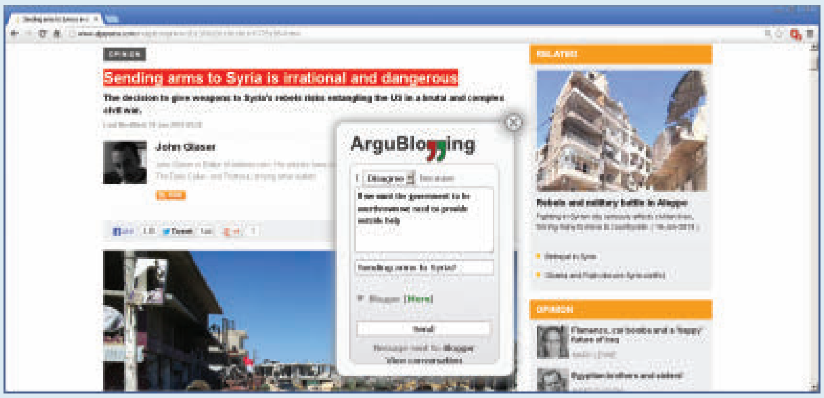
Arvina can take on a multi-agent system populated by agents representing authors whose opinions are available on the Argument Web. It is thus possible to question the participants of, say, past Moral Maze radio programs about their opinions, and agents representing participants can be added to a discussion and questioned about what they think about the topic discussed during the program. An agent then answers by giving the opinions originally expressed during the broadcast by a particular participant. In this way people are able to question any opinion expressed on the Argument Web, whether originally added through OVA, ArguBlogging, or other tools.

Argument blogging. A final example of how argumentation technologies based on the Argument Web facilitates online debate concerns blogging, a highly popular form of online communication. If one wants to reply to an opinion presented somewhere on the Web in a blog, the usual way is to provide a simple hyperlink to the article or page in which the opinion is expressed. However, the resulting structure of supporting and competing opinions is easily lost due to lack of semantic information in the links.

We built a simple tool called ArguBlogging, for Argument Blogging,[10] to improve rational debate through this popular form of expressing opinions online. It ensures that, if desired, opinions on blogs and other webpages can be linked through the underlying infrastructure of the Argument Web, allowing users to agree or disagree with opinions anywhere on the Web, simultaneously posting it to their blog (connecting to two popular blogging platforms, Blogger and Tumblr) and adding it to the Argument Web; Figure 5 is the ArguBlogging window (or widget) rendered on a webpage, providing options for responding.

A typical blog post through ArguBlogging contains an "argue" button that, clicked, brings the widget onscreen. It allows users to respond to the blog post as a whole through ArguBlogging and is similar to the "like" button on Facebook and "share" button on Twitter." This way, the ArguBlogging tool uses familiar interaction styles to enable critical argumentation on the Web.

Figure 5. The ArguBlogging widget.



Sending arms to Syria is irrational and dangerous

The decision to give weapons to Syria's rebels risks entangling the US in a brutal and complex civil war.

John Glaser

If we want the government to be overthrown we need to provide outside help

Sending arms to Syria?

Message sent to Blogger

View conversation

Rebels and military battle in Aleppo

Evaluating the Argument Web

The kind of continuous evaluation needed by the Argument Web involves usability in two distinct domains—public and academic—and expressivity and computational flexibility through metrics drawing on formal, computational methods, as well as on more pragmatic engineering principles.

In terms of raw usage through compatible tools (such as Rationale and Araucaria), the Argument Web today includes tens of thousands of users worldwide. The native applications (such as OVA) released in the past few years naturally involve fewer, but the core AIFdb includes more than 11,000 argument components, the second largest semantically rich argumentation dataset (after Debategraph), including in seven languages; for example, the Archelogos repository is fully imported into the Argument Web. More significant, the infrastructure and analysis tools have been used in at least seven research projects involving 10 labs in France, Poland, the U.K., and the U.S., each with different foci, including generative and analytical, multilingual and monolingual, and dialogical and monological.

In terms of expressivity, the Argument Web balances well-defined formal properties and pragmatic solutions to engineering problems; for example, the underlying ontology is demonstrably more expressive than one of the foremost formal accounts of defeasible argumentation, ASPIC+,[2] but a well-defined subset can be used to induce abstract frameworks on which evaluation can be performed.[16] Ongoing work in discourse analysis[5] demonstrates that even rhetorically and linguistically sophisticated maneuvers (such as ad hominem argumentation) can be captured in the Argument Web. At the same time, formal description-logic analysis of the AIF ontology[13] shows powerful abstractions can support advanced search and evaluation of arguments involving, say, specific schemes of presumptive reasoning.

The Argument Web's ultimate objective is to improve the quality of online argument and debate. Keeping in mind that evaluation of natural argument is philosophically thorny, it is possible to propose objective metrics through which to assess natural arguments, including, say, consensus about which arguments are best (one might disagree with an argument while still appreciating its merits); exhaustiveness, speed, volume of content, signal to noise, and structural complexity; argument richness (in terms of the range of argument types used); and dialogue richness (range of dialogical moves used).

Conclusion

The Argument Web represents the first technology linking debate, disagreement, and argument structures from a variety of tools applied in different domains. The approach has strong potential both academically and practically. Along with developer and user interest and increasing Argument Web resources, the academic community gains access to a valuable resource that, particularly for computer scientists, can function as a testbed for new theories of argument-acceptability applications and as a rich dataset with which to deploy new applications; for linguists and philosophers, it offers a unique corpus of discourse activity, replete with detailed annotation and commentary.

By solving theoretical problems involving how argument structures can be navigated and extended through dialogical processes, the Argument Web opens up a new class of application in which intuitive, dialogically based interfaces (such as Arvina and ArguBlogging) can be used to explore and improve large-scale debates. There is evidence[19] that debate is a good way to navigate and support engagement with complex issues involving disagreement (such as abortion, climate change, and military intervention) that are not just important but that define our time. By supporting and facilitating engagement in debates (otherwise daunting, leading even to disengagement and disempowerment) the Argument Web promises to play not only a technological but an important societal role as well.

Debate is a good way to navigate and support engagement with complex issues involving disagreement (such as abortion, climate change, and military intervention) that are not just important but that define our time.

References

1. Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American 284*, 5 (2001), 28–37.
2. Bex, F., Modgil, S., Prakken, H., and Reed, C. On logical reifications of the Argument Interchange Format. *Journal of Logic and Computation.* Aug. 2012; doi: 10.1093/logcom/exs033
3. Bex, F. and Reed, C. Dialogue templates for automatic argument processing. *Frontiers in Artificial Intelligence and Applications 245* (2012), 366–377.
4. Bizer, C., Heath, T., and Berners-Lee, T. Linked data: The story so far. *International Journal on Semantic Web and Information Systems 5*, 3 (2009), 1–22.
5. Budzynska, K. and Reed, C. The structure of ad hominem dialogues. *Frontiers in Artificial Intelligence and Applications 245* (2012), 410–421.
6. Chesñevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., and Willmott, S. Towards an Argument Interchange Format. *Knowledge Engineering Review 21*, 4 (2009), 293–316.
7. De Liddo, A., Sándor, Á., and Buckingham Shum, S. Contested collective intelligence: Rationale, technologies, and a human-machine annotation study. *Computer Supported Cooperative Work 21*, 4–5 (2012), 417–448.
8. de Moor, A.D. and Aakhus, M. Argumentation support: From technologies to tools. *Commun. ACM 49*, 3 (Mar. 2006), 93–98.
9. Dung, P.M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n–person games. *Artificial Intelligence 77*, 2 (1995), 321–357.
10. Lawrence, J., Snaith, M., Bex, F.J., and Reed, C. Demonstration papers on the AIFdb, ArguBlogging, Arvina, and TOAST. *Frontiers in Artificial Intelligence and Applications 245* (2012), 511–516.
11. Okada, A., Buckingham Shum, S.J., and Sherborne, T., Eds. Knowledge Cartography: Software Tools and Mapping Techniques. *Advanced Information and Knowledge Processing.* Springer, London, 2008.
12. Pinkwart, N., Ashley, K., Lynch, C., and Aleven, V. Evaluating an intelligent tutoring system for making legal arguments with hypotheticals. *International Journal of Artificial Intelligence in Education 19*, 4 (2009), 401–424.
13. Rahwan, I. Mass argumentation and the Semantic Web. *Web Semantics 6*, 1 (2008), 29–37.
14. Rahwan, I., Zablith, F., and Reed, C. Laying the foundations for a World Wide Argument Web. *Artificial Intelligence 171*, 10–15 (2007), 897–921.
15. Reed, C., Bex, F., Lawrence, J., and Snaith, M. DEMO: The Argument Analysis Wall. In *Proceedings of Digital Futures: The Third Annual Digital Economy All Hands Conference* (Aberdeen, U.K., 2012).
16. Snaith, M. and Reed, C. TOAST: Online ASPIC+ implementation. *Frontiers in Artificial Intelligence and Applications 245* (2012), 509–510.
17. van Eemeren, F.H. and Grootendorst, R. *A Systematic Theory of Argumentation: The Pragma-Dialectical Approach.* Cambridge University Press, Cambridge, U.K., 2004.
18. Walton, D.N., Reed, C.A., and Macagno, F. *Argumentation Schemes.* Cambridge University Press, Cambridge, U.K., 1995.
19. Yuan, T., Moore, D., Reed, C., Ravenscroft, A., and Maudet, N. Informal logic dialogue games in human-computer dialogue. *Knowledge Engineering Review 26*, 3 (2011), 159–174.

Floris Bex (f.j.bex@rug.nl) is a researcher and instructor at the University of Groningen, Groningen, The Netherlands.

John Lawrence (johnlawrence@computing.dundee.ac.uk) is a Ph.D. student at the School of Computing of the University of Dundee, Dundee, U.K.

Mark Snaith (marksnaith@computing.dundee.ac.uk) is a postdoctoral research assistant at the School of Computing of the University of Dundee, Dundee, U.K.

Chris Reed (c.a.reed@dundee.ac.uk) is a professor at the School of Computing, University of Dundee, Dundee, UK.

DOI:10.1145/2500892

Keywords in the ACM Digital Library and IEEE Xplore digital library and in NSF grants anticipate future CS research.

BY APIRAK HOONLOR, BOLESLAW K. SZYMANSKI, AND MOHAMMED J. ZAKI

Trends in Computer Science Research

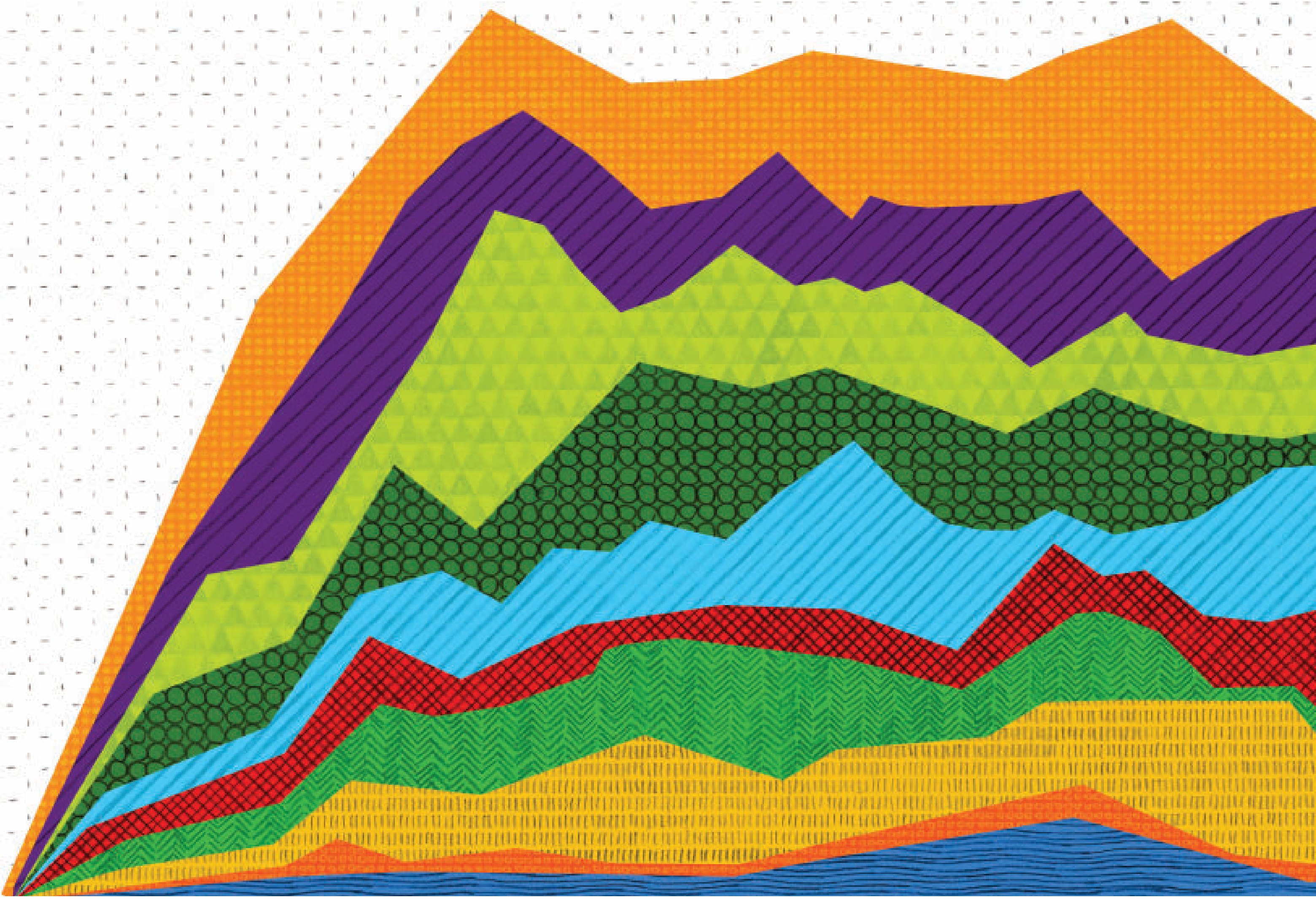
COMPUTER SCIENCE IS an expanding research field driven by emerging application domains and improving hardware and software that eliminate old bottlenecks even as they create new challenges and opportunities for CS research. Accordingly, the number of research papers published in CS conferences and journals has been increasing rapidly for the past two decades. With growing emphasis on externally funded research in most universities, scientific research is increasingly influenced by funding opportunities. Although many funded programs are developed in close collaboration with leading researchers, we aim here to identify more precisely relationships between funding and publications related to new topics.

Trend analysis has long been researched and applied to many types of datasets, from medical[17] to weather[15] to stock markets.[5] Many publications track research trends, analyze the impact of a particular paper on the development of a field or topic, and study the relationships between different research fields. The Web of Science[22] has collected data since 1900 on nearly 50 million publications in multiple scientific disciplines and analyzed it at various levels of detail by looking at the overall trends and patterns of emerging fields of research and the influence of individual papers on related research areas. Over the past decade, besides the Web of Science, studies have also investigated the overlap and evolution of social communities around a field or a topic. Rosvall and Bergstrom[18,19] explored methods and visualizations for scientific research and analyzed the impact of each research area quantified by the collective cross-disciplinary citations of each paper. Porter and Rafols[16] analyzed citation information to find evidence of collaboration across fields in scientific research. Other examples are network models for studying the structure of the social science collaboration network[13] and women's authorship of CS publications in the ACM digital library.[3]

Several studies have focused on challenges, directions, and landscapes in specific CS fields[2,7] and on

» key insights

- **A burst of new keywords in grants generally precedes their burst in publications; less than one-third of new keywords burst in publications first, reflecting the importance of funding for success of new CS fields.**
- **A typical scientist's research focus changes in roughly a 10-year cycle and often includes a once-in-a-career shift, likely in response to evolving technology creating new CS fields.**
- **CS continues to experience continuous and fundamental transformation; for example, in the past two decades, new topics arose within the Internet research cluster, while some previously popular topics (such as mathematical foundations) decayed.**



specific CS topics.[8,21] In this article, we are interested in learning about the evolution of CS research. We collected data from 1990 to 2010 on proposals for grants supported by the U.S. National Science Foundation[14] and on CS publications in the ACM Digital Library[1] and IEEE Xplore digital library.[11] We analyzed research communities, research trends, and relations between awarded grants and changes in communities and trends, as well as between research topics. We found if an uncommonly high frequency of a specific topic is included in publications, the funding for the topic usually increases. We also analyzed CS researchers and communities, finding only a small fraction of authors attribute their work to the same research area for a long time, reflecting an emphasis on novelty (new keywords) and frequent changes in academic research teams (a stable core of faculty and turnover of students and postdocs). Finally, our work highlights the dynamic CS research landscape, with its focus constantly moving to new challenges due to new technological developments. CS is an atypical academic discipline in that its universe is evolving so quickly, at a speed unprecedented even for engineering. Naturally, researchers follow the evolution of their artifacts by adjusting their research interests. We attempt to capture that vibrant coevolution here.

We used the ACM, IEEE, and NSF datasets from which we collected data on publications from 1990 to 2010.[a] For the ACM dataset, we extracted the number of papers listed in top categories of the 1998 ACM Computing Classification System, or CCS.[b] The ACM dataset included authors, title, abstract, year published, publication venue, author-defined keywords, and ACM classification categories for each of the 116,003 articles. We used the ACM CCS and author-defined keywords to respectively study the broader and static versus the finer and dynamic views of the CS landscape and trends. In another analysis, we used only the author-defined keywords to identify the relationships between researchers, yielding smaller research groups than if we had used just ACM CCS alone.

The IEEE Xplore dataset included similar information but lacked a topic classification like the ACM CCS. Instead, we used 408 research topics included in 16 Wikipedia articles on CS research areas identified in the main Wikipedia CS article[23] to classify 458,395 papers in the IEEE dataset. For the NSF dataset, we retrieved titles, start dates, and abstracts of 21,687 funded grant proposals.

For the ACM and IEEE datasets, we created two data indexes—authors and their publication venues and papers and their keywords/topics—finding, in the analyzed period, the number of publications grew approximately 11% yearly over those 20 years. To create research topic networks, we made each

a NSF records before 1990 were incomplete (such as lacking abstracts), but only 10% of publications in the ACM and IEEE datasets were published before 1990, so our time range covers nearly all publications in those datasets.

b We excluded the "general literature" category because CCS (http://www.acm.org/about/class/1998/) includes too many non-research topics (such as biography and reference).

topic a node and connected two nodes with a weighted edge representing the number of abstracts that mention both adjacent topics.

Using sequence mining,[24] network extraction and visualization,[18] bursty words detection,[12] clustering with bursty keywords,[c,10] and network evolution,[6] we investigated changes over time in the CS research landscape, interaction of CS research communities, similarities and dissimilarities between research topics, and the impact of funding on publications and vice versa.

Results and Discussion

Landscape of CS research. We looked at the evolution of the CS research landscape 1990–2010 (see Figure 1). Many ACM records 2009–2010 (collected during the spring of 2011) did not have ACM classification categories and thus were excluded from our study, causing the drop off of records in the last two years in Figure 1. (There is no such drop in Figure 4 because every record included a publication venue.) For ACM, except for 2009 and 2010, publications in each category increased year over year, but after 1994 the fraction of publications in the "mathematics of computing" category shrank considerably. The author-defined keywords contributing to the drop were control theory and logic. We attribute the drop to a shift of focus from general issues to challenges specific to an area with which such publications are increasingly associated. For the remaining categories, the fastest growing were publications in information systems. The most frequently used author-defined keywords were Internet-related (such as XML, Internet, Web services, and semantic Web). Likewise, the IEEE dataset showed the fastest-growing research area was information science and information retrieval.

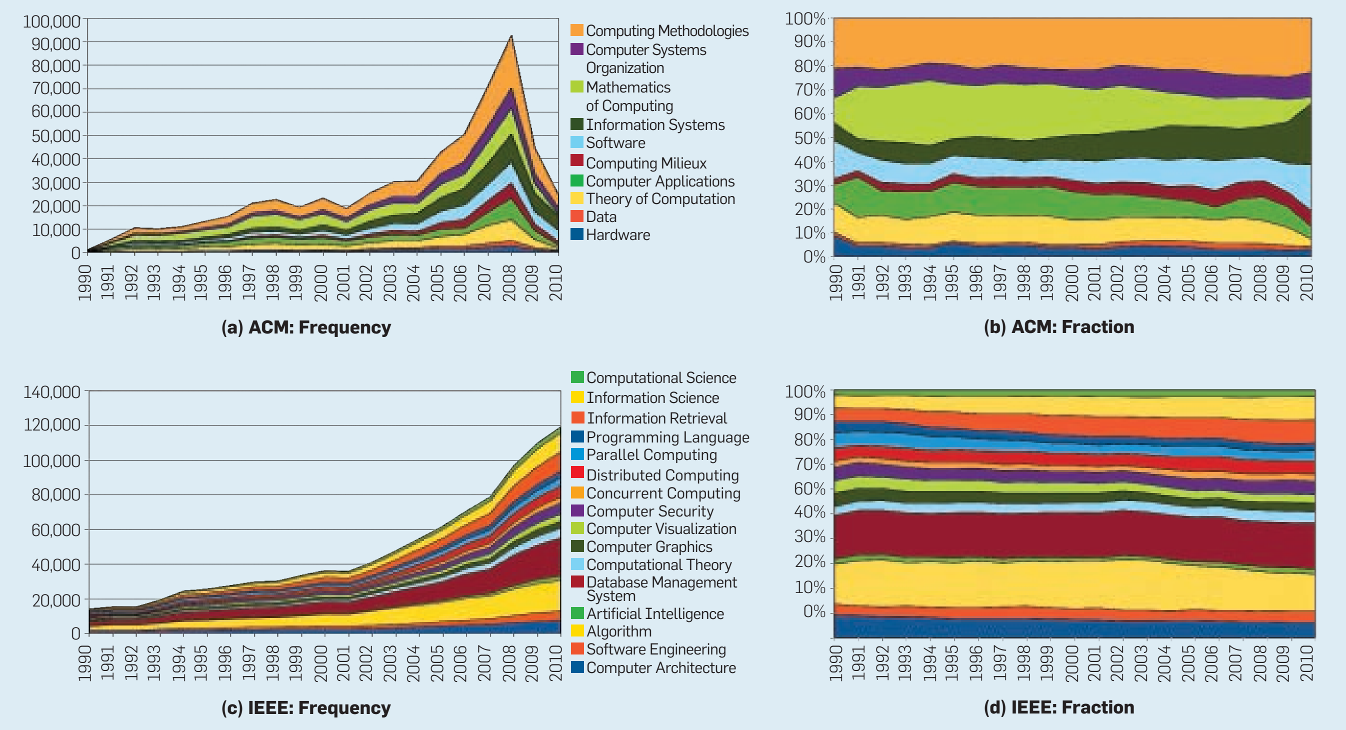
To better see the impact of information systems, we extracted the top 25 research topics from ACM and IEEE and quantified the results in two ways: document frequency (DF) and term frequency inverse document frequency (TFIDF). For term/keyword k, DF is the number of documents including it; TFIDF is the sum of the weights over all documents, where the weight of k in document d is defined as

$$\frac{n_{k,d}}{\sum_{w\in d} n_{w,d}} \cdot \log \frac{|D|}{|j : k \in d_j|}$$

where $|D|$ is the number of documents and $n_{k,d}$ is the number of times k appears in d; see Hoonlor et al.[9] for detailed results. Most publications in collaboration, data mining, information retrieval, machine learning, privacy, and XML appeared 2000–2010 and showed noteworthy trends in CS research. The terms Internet and World Wide Web did not appear in any publication until 1995, but the related topics were present since early 1990. During the period 1990–1997, 376 NSF grants and nine IEEE papers mentioned NSFNET in their abstracts, but only two ACM papers included it as a keyword. Other terms (such as net, prodigy, point-to-point, and internetworking) also appeared in the NSF dataset before 1995. More-

c The term "bursty keywords" in this context refers to keywords appearing with uncommonly high frequency during some intervals; such intervals may include multiple spikes of a keyword's frequency.

Figure 1. Two views of CS research, 1990–2010, based on the ACM (a) and (b) and IEEE (c) and (d) datasets; frequency is number of publications on each topic each year; fraction is the percentage of publications on each topic each year.



over, prodigy was bursty over the period 1991–1992 and TCP/IP over the period 1990–1993.

TFIDF and DF values showed the rise of information system contributed to the general interest in data mining, information retrieval, and Web-related topics, whereas mathematics of computing continued to decrease year over year during the same period, with logic and control theories contributing most to the decline.

Figure 2 includes the research topic subnetworks culled from the ACM dataset by the Map Generator software package[4] for the security and the multimedia subnetworks found in 1995 and for the World Wide Web and the Internet subnetworks found in 2001. In 1995, Web was used as a keyword associated mostly with multimedia and information visualization, whereas information retrieval was used mostly with Internet. However, by the early 2000s, Web was used mostly with data mining and information retrieval, while Internet was used mostly with network, protocol, and routing. Since 2005, privacy and security have become important in the Web context, while semantic Web, Web 2.0, Web service, and XML became major Internet topics.

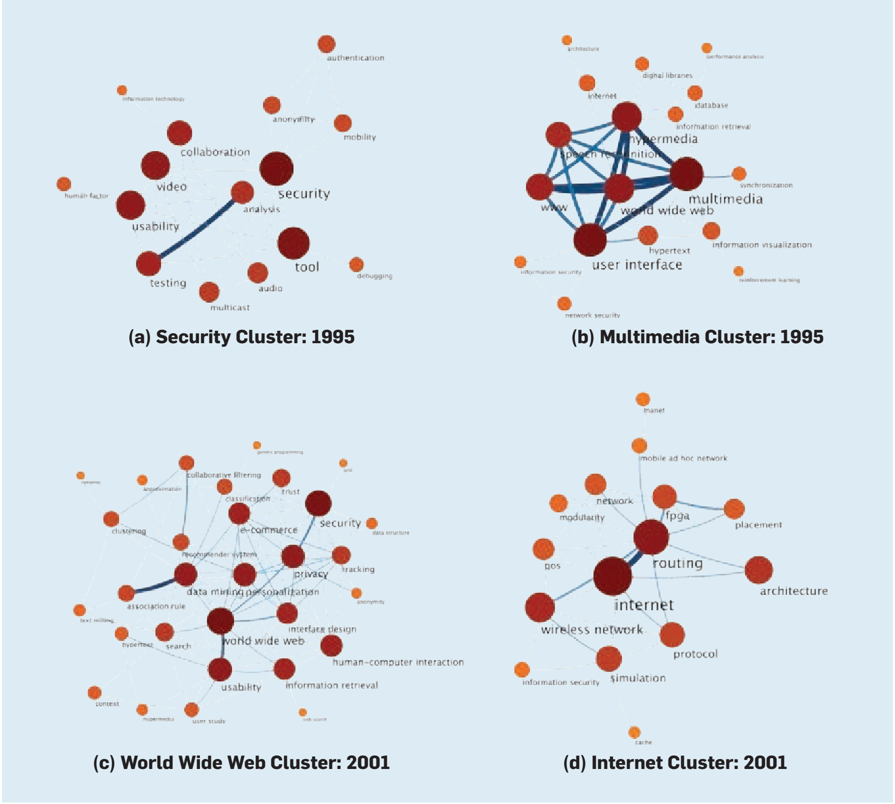
Bursty-period analysis. To evaluate the influence of research funding on publications and vice versa, we extracted from ACM,[d] IEEE, and National Science Foundation datasets the bursty periods of author-defined keywords based on the burstiness score for a time period[12] defined as

$$Burst(w,t) = \left(\frac{|d_t : w \in d_t|}{|d : w \in d|} - \frac{1}{T} \right)$$

where w is the keyword/topic of interest, t is a time period, d_t is a document created during time t, d is any document, and T is the total time over which all documents were created. The burstiness score measures how often w is in t compared to its occurrence in T. A positive score implies w appears more often during the "bursty period" t than over the total time T. We recovered the maximal segments of burstiness scores in the sequence of documents using the linear-time maximum sum algorithm;[20] each segment with positive score corresponds to a bursty period.

d Since only ACM records are classified through CCS, we do not use that classification here.

Figure 2. Research clusters, or subnetworks, in 1995 and 2001; edge thickness represents strength of interaction.



(a) Security Cluster: 1995

(b) Multimedia Cluster: 1995

(c) World Wide Web Cluster: 2001

(d) Internet Cluster: 2001

For each pair of datasets, we analyzed in which one a keyword's bursty period begins first and then how long it takes for the keyword to become bursty in the other. For keywords with more than one bursty period, we also looked at their burstiness score in each bursty period, then tabulated the percentage of cases in which the later burstiness scores increased, decreased, or was unchanged.

For an ACM-NSF pair, if a keyword became bursty in ACM data first, it became bursty in NSF 2.4 years later on average; in the reverse case, the average delay was 4.8 years. The longer delay shows if NSF initiates a new area, the increase in publications is delayed by the time researchers need to obtain grants and start research leading to publication. If the keywords were bursty in both datasets, in 75% of such cases the keyword became bursty in the NSF dataset before it became bursty in the ACM dataset, showing NSF funding often increases interest in the supported areas. For another 16% of cases, it was reverse; examples of bursts appearing first in the NSF dataset are data mining and search engine, becoming bursty in 1999 for NSF and in 2000 for ACM. The reverse included bioinformatics (2003 in ACM and 2004 in NSF) and semantic Web (2004 in ACM and 2006 in NSF).

For an IEEE-NSF pair, a keyword first bursty in IEEE became bursty in NSF 3.4 years later on average; in the reverse case, the average delay was 5.7 years. The difference between these two delays and its causes are the same as in the ACM dataset. Yet both delays are one year longer than in the ACM-NSF pair, resulting, we conjecture, from a larger ratio of computer engineering topics in IEEE than in ACM and presumably to a larger fraction of support for IEEE publications from non-NSF sources.

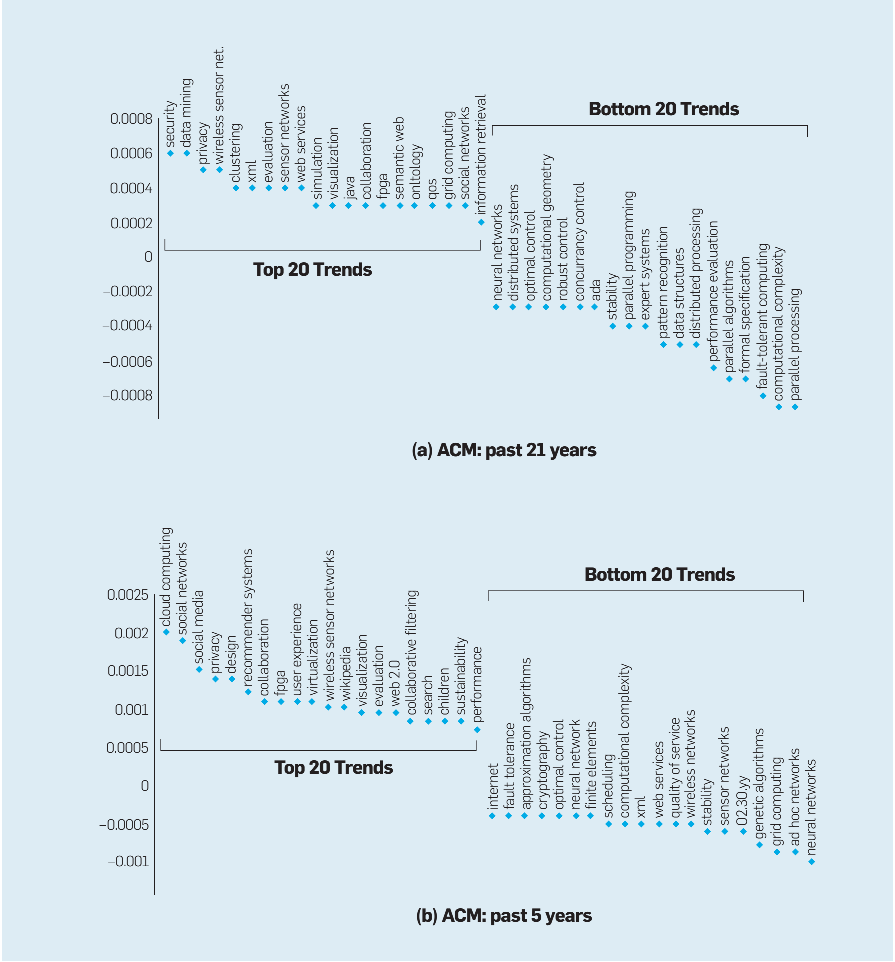
In two-thirds of the cases where a keyword was bursty in both the NSF and the IEEE datasets, it became bursty in the NSF dataset first, consistent, again, with the ACM dataset. The other cases were evenly split between the reverse and the concurrent appearance of burstiness in both datasets. Only Internet (in 2000) and telecommunications (in 1995) became bursty at the same time in both datasets. Keywords bursty first in the IEEE dataset included real-time database (1994 versus 1999 for NSF), procedural programming (1992 versus 1993), and neuro-biological (1996 versus 2001). Peer-to-peer network was bursty in IEEE 2003–2010 but never in the NSF dataset, possibly indicating the corresponding challenges were funded mostly by non-NSF sources; see Hoonlor et al.[9] for more detailed bursty-period comparisons.

We also analyzed the NSF dataset versus ACM and IEEE datasets and vice versa. For each such pair and each year 1990–2010, we searched for the year in which the number of entries changed compared to any of the previous four years in the first database. For each change, we searched in the other dataset for a change in any of the next four years. The relative change values ranged from −0.5 to 0.5, which we grouped into bins of size 0.1. We counted the frequency of the change in one dataset followed by a change in the other.

For the NSF dataset versus either the ACM or the IEEE dataset, a 10% or greater increase in the number of NSF grants awarded for a given topic from the previous few years was followed by an increase (with 75% probability) in the number of published papers on the topic of at least 10% in the next three years and 20% in the next four years. Topics with such an increase included data mining, information extraction, and wireless network. On the other hand, an increase of 10% in the number of published papers in a given topic in the ACM dataset was followed with 75% probability of an increase (usually less than 10%) in the number of NSF grants awarded on the same topic; examples were e-government, groupware, and knowledge management.

For a keyword with multiple bursty periods in the NSF dataset, the following bursty period had a higher/lower/equal burstiness score in 37%/51%/12% of the cases. For the IEEE dataset, it was 29%/64%/7%, respectively, and for the ACM dataset, it was 12%/85%/4%. However, for interleaved or overlapped bursty periods in the NSF and IEEE datasets, if the bursty period appeared first in the IEEE dataset, the following NSF bursty period had a higher/lower/equal burstiness score in 31%/22%/47% of the cases. In the reverse case, it was 36%/10%/55%. The same analysis of the ACM and NSF datasets showed the following NSF bursty period had higher/lower/equal burstiness score for 38%/14%/48% of the cases; in the reverse case, for the following ACM bursty period, the numbers were 8%/8%/84%.

Figures 3a and 3b. Top 20 and bottom 20 trends 1990–2010 and 2006–2010 from the ACM and IEEE datasets.

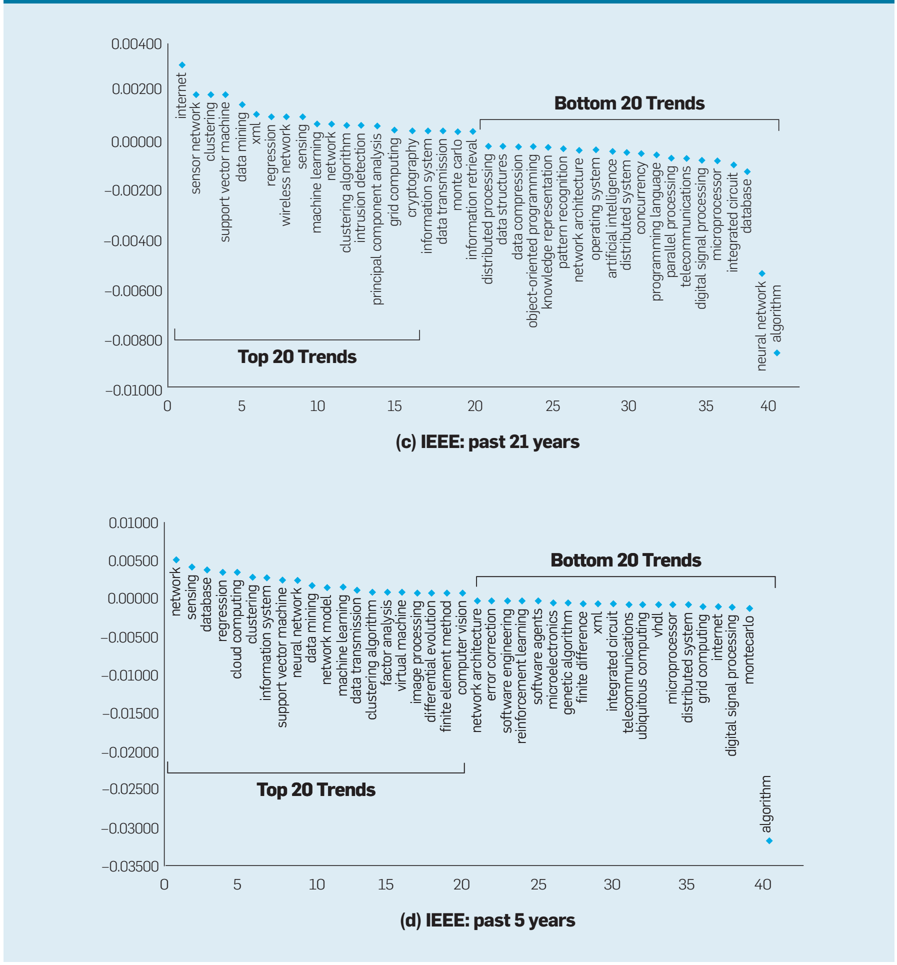


The reason for a large percentage of equal burstiness scores is that a bursty period in one dataset was often a subset of the bursty period in another. Burstiness scores tend to decrease in the periods following a bursty period in the NSF dataset. Since novelty is prized so highly in publications, authors tend to stress new aspects of their work in abstracts and keywords, contributing to the observed pattern. Yet during an NSF burstiness period, publication burstiness scores were more likely to increase than decrease, confirming sustained NSF funding is essential for maintaining interest in a given topic.

Further analysis identified keywords associated with each bursty period that burst together; for example, wireless sensor networks are temporally related to simulation, security, and clustering in the order of bursty periods. This order corresponds to the temporal evolution of the area that initially focused on simulation of networks, then on security, and finally on clustering algorithms. The analysis also revealed data mining is more broadly used than information retrieval. The former is used with computational science, Web mining, time series mining, and security; the latter is used mainly with Web-related topics. Text mining is temporally related to both information retrieval and data mining.

Multiple bursty periods for a keyword include interesting temporally correlated terms. For example, there were three bursty periods for the keyword "scheduling": 1990–1991, 1999, and 2001–2006. In the bursty period of 1999, scheduling correlated (listed

Figures 3c and 3d. Top 20 and bottom 20 trends 1990–2010 and 2006–2010 from the ACM and IEEE datasets.



in the order of burstiness ranking) with genetic algorithms, parallel processing, performance evaluation, embedded systems, approximation algorithm, multimedia, quality of service, optimization, and heuristics. In the period 2001–2006, such keywords, listed in the same order, were approximation algorithms, multimedia, online algorithms, real time, embedded systems, fairness, multiprocessor, quality of service, and genetic algorithms. Hence, initially, both real-time systems and parallel processing were related to scheduling, later expanding to genetic algorithms and embedded systems. In the last few years of its bursty periods, scheduling also correlated with multimedia, online algorithm, and fairness. An alternative look at such links can be done through the co-referenced document frequency instead of the burstiness score.[9]

Trend analysis. Here, we analyze research trends through the linear regression trend line and changing popularity of topics based on the fraction of papers including a given keyword in each year. We generated a trend line for each keyword fraction and used its slope for ranking. We fit the trend lines to data from the preceding two to six years in order to predict keyword fractions for the following year.

In all datasets, we observed that if a trend based on two years of data had a positive slope, or the fraction of publications increased from the previous year to the current year, then the subsequent year fraction declined. We also used the trend line based on the NSF dataset to predict fractions for the following year in the ACM and IEEE datasets. The results show the trend line is a poor predictor, as is using ACM and IEEE trends to predict the number of grants awarded by NSF; the accuracy of all these models was less than 50%.

Figure 3 includes the top 20 up and top 20 down trends for the period 1990–2010 and for the period 2006–2010 for ACM and IEEE. Unlike the ACM dataset, the IEEE dataset did not show a significant decrease between the top and the bottom trends because research topics appear in the abstracts longer than do author-defined keywords. Moreover, we used the CS conferences listed in Wikipedia[23] to categorize each paper in IEEE and ACM; see Hoonlor et al.[9] for the complete list of categories. We could not statistically compare the growth in different areas due to vast differences in the number of conferences in each field and number of papers published in each conference. Nevertheless, Figure 4 indicates growth of approximately 11% in publications for most CS topics year over year, with each topic representing a set of CS conferences;[e] when a conference covered two topics, then papers published in the conference were indexed in both topics.

Network of CS research. Since we looked back over the period 1990–2010, we were also able to monitor when connections between two fields occurred or changed. We extracted two sets of keywords: those never appearing together in the same article and those appearing together in at least some specified number of articles each year. For the IEEE dataset, keeping algorithm as a node greatly reduced the degree of separation between other research topics and created a central node, or one with the highest total weight of its edges, dominating other research topics. We performed the network analysis on the algorithm topic first, then removed the algorithm node from the network because the term was used in almost every CS research paper to describe how data is processed.

During the period 1990–2010, algorithm, database, and neural network were the most frequent CS research topics; 311 other CS research topics were also mentioned, along with algorithm at least once, with 78 of them persistent; that is, they co-appeared with algorithm every year from 1990

e In contrast, Figure 1 uses ACM classification and IEEE Xplore keywords; even ACM records missing ACM classification terms are represented here since each record includes publication information.

to 2010. Of 408 CS research topics, 286 were mentioned with database, but only 32 were persistent topics.[f] Meanwhile, 254 topics appeared with neural network, with only 39 persistent. Besides the three most frequent topics, 11 others had persistent connections with multiple research topics every year 1990–2010, including programming language, artificial intelligence, clustering, image processing, computer vision, network, distributed system, pattern recognition, robotics, software engineering, and integrated circuit. Also during 1990–2010, 87 other research topics, including image analysis, data transmission, and operating system, were linked with up to three of the mentioned 14 topics.

In ACM networks using author-defined keywords, no persistent link appeared 1990–2010, confirming our earlier observation that while a certain research topic may be important enough to be mentioned in an article's abstract, it may not represent the article's key research contribution. Another example of lack of link persistence is the neural network node in both IEEE and ACM networks. In the former, neural network was a central node in almost every year. Yet in ACM networks, it never achieved this status. Lack of link persistence is also evident for algorithm and database topics. In the early 1990s, user interface, scheduling, and multimedia were associated with many CS research fields. However, in the late 1990s, interest shifted to the Web, information retrieval, and computer-supported cooperative work. Throughout the 2000s, the areas most connected to others were design, usability, and security; the mid-2000s saw strong interest in sensor network and later in wireless sensor network.

We performed clustering on the yearly network of keywords in the ACM dataset in which a keyword can appear in multiple clusters; using the clusters, we measured the similarity between keywords k and a as

$$\frac{\text{Number of clusters with } a \text{ and } k}{\text{Number of cluster with } a}$$

We found a list of terms clustered together with network connectivity in the period 2006–2010 though not connected in at least 1% of the documents.[24] We examined the top 10 frequently used keywords at various degrees of separation; see Hoonlor et al.[9] for results. During the period 2006–2010, simulation was clustered with many keywords in database research, including integration, data warehouse, and relational database, although they were either not used or used only rarely by authors to describe their research in simulation. Simulation was instead clustered with information retrieval, feature selection, and filtering, as well as with other topics related to data mining, machine learning, and artificial intelligence, but was not used directly to describe the same research project often enough. Data mining was rarely used to describe research related to mobile networks and its related research topics.

CS researchers. We used the cSpade sequence mining algorithm[24] to analyze sequences of publications in the same major research category by the same author, requiring at most a one-year gap in publication dates and appearance in at least 1% of documents. We recorded the maximum length of publication sequences in the same category and measured the percentage change in the number of publications of a given author after the first year in each category. From all the authors whose publications were in the same categories, we calculated the half-life time (it took for the number of authors who continued publishing papers in the category to reduce by half). For the first analysis, we used the ACM CCS to identify major research categories, then performed the same analysis using the list of conferences under six CS categories, finding the rates of publication growth differed in each category; see Hoonlor et al.[9] for details.

We found a relatively short half-life time, as well as a significant first-year drop rate, especially for computer application, computing milieu, and data keywords, indicating the authors in these categories were either only briefly involved in multiple research topics or

An increase of 10% in the number of published papers in a given topic in the ACM dataset was followed with 75% probability of an increase (usually less than 10%) in the number of NSF grants awarded on the same topic.

f Top five persistent topics in the database research cluster were relational database, distributed database, database management, query language, and database design; for the neural network research cluster, they were pattern recognition, regression, supervised learning, reinforcement learning, and robotics.

only briefly collaborated with someone else from these categories. Researchers in computer systems organization, computing methodologies, and information systems tended to stay active in these categories for a longer time. Moreover, we found it difficult for researchers to publish in artificial intelligence and programming language year after year, unlike in, say, human-computer interaction. Researchers in human-computer interaction remain active the longest, followed by researchers in computer architecture; see Hoonlor et al.[9] for details.

Note while researchers can continue to publish in one area for a long time, the area itself evolves and may cover different topics during different time periods; for example, human-computer interaction focused mainly on interaction design, visual design, and computer-supported cooperative work in the 1990s and augmented reality, computer vision, human factors, and ubiquitous computing in the early 2000s, finally shifting to social media, learning, computer-mediated communication, and tangible user interface in the late 2000s.

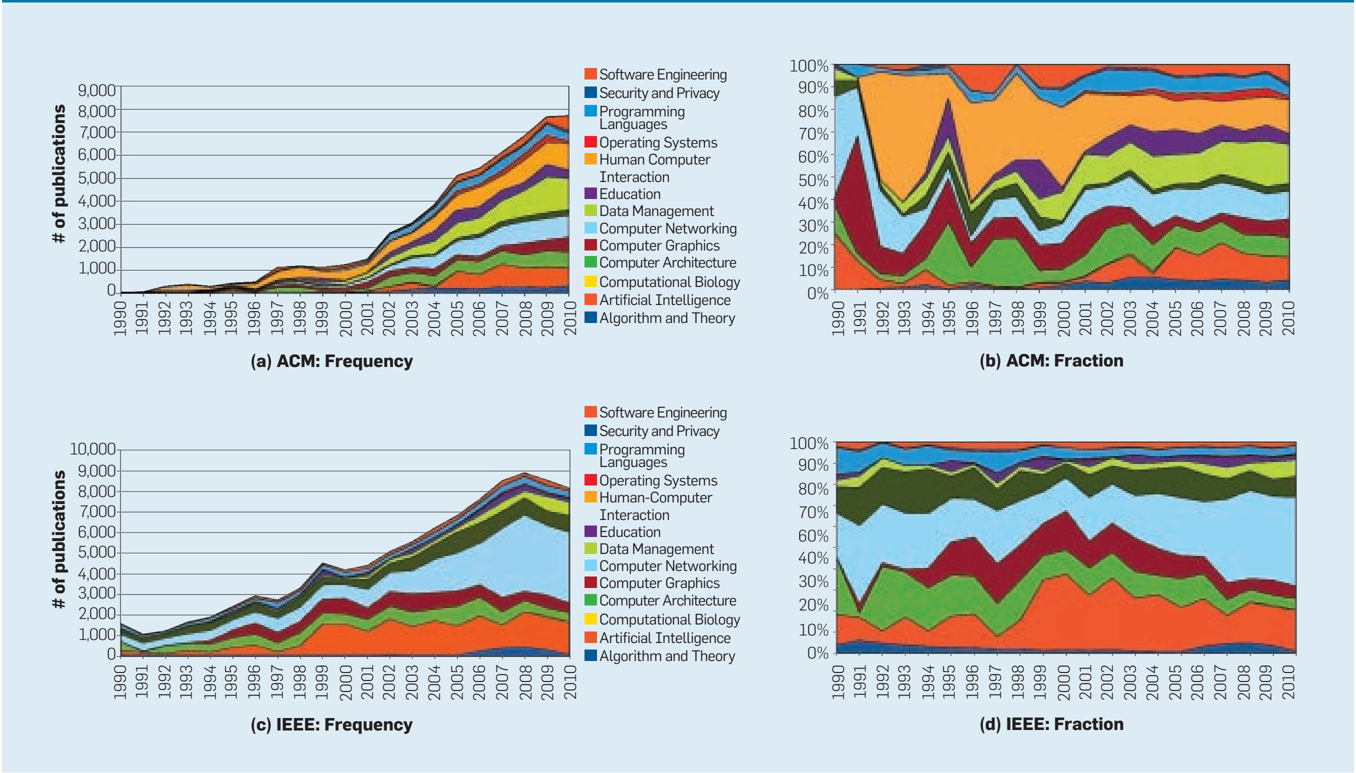
Investigating further, we selected four prominent CS researchers, analyzed their publications, and discussed the results with them. Jack Dongarra of the University of Tennessee, Knoxville, is renowned for developing high-performance linear algebra software packages and systems, though his interests have evolved over time. In the 1980s, for example, he focused on parallel algorithms for linear equation routines and linear algebra subprograms. In the early 1990s, he focused on parallel solutions for eigenvalue problems and numerical software libraries for high-performance systems. From the late 1990s to the 2000s, he focused on high-performance linear algebra packages for multicore systems. More recently, he has also focused on performance of grid computing. Overall, his research interests have evolved continuously in response to challenges created by new computer technologies.

Another researcher in this area, Francine Berman of Rensselaer Polytechnic Institute, Troy, NY, characterized her work in 1980s as "top-down mathematical modeling" of mapping and scheduling problems. In the early 1990s, her papers used such keywords as data-driven, performance, and algorithms. From the late 1990s to the mid-2000s, she focused on grid computing from a "bottom up" perspective: application-level scheduling/rescheduling, job distribution, and performance. She described this evolution as a broadening and branching approach. Since 2003, she has made a major shift to large-scale cyberinfrastructure and data preservation.[g]

In the early 1990s, George Cybenko of Dartmouth College, Hanover, NH, studied high-performance computing and classification by neural networks. In the late 1990s, he shifted to mobile agents, mobile networks, and simulations. In the early 2000s, he focused on target tracking, analyzing data, and extracting information from the Web and from wireless networks. Since 2002, he has investigated privacy and security issues, including cybersecurity, saying he investigates each subject "in five-year (more or less) phases" then "discovers an open field often related to previous work." One exception was a major shift in 1992 when moving from one university to another.

g Cyberinfrastructure and data preservation did not show up as her keywords because the relevant publications were too new to be in our databases.

Figure 4. Landscape of CS research fields, based on conferences 1990–2010, for the ACM and IEEE datasets, including raw numbers (frequencies) and percentage of publications for each keyword each year.

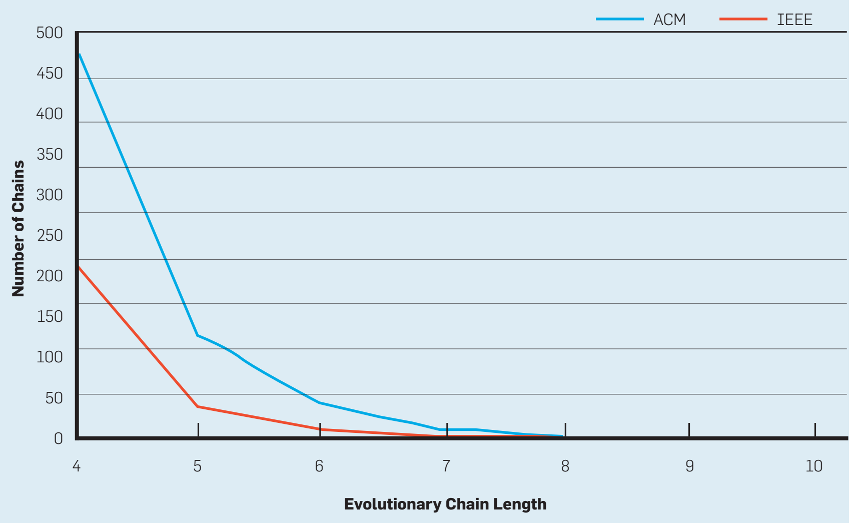


As a final example, James A. Hendler of Rensselaer Polytechnic Institute, Troy, NY, has worked in artificial intelligence since the late 1980s. His major shift was from planning and Web intelligence to the semantic Web. From the late 1980s to early 1990s, his work focused on planning in artificial intelligence and later on agents, real-time systems, and Web technology. In the 2000s, he focused mainly on the semantic Web and for the past few years also on large data and social networks.

Overall, CS faculty research interests typically evolve every five to 10 years by broadening their scope and branching into new applications, as well as responding to technological innovation. Less frequently, perhaps once in a career, there is a major shift to a new area.

Communities of CS researchers. Using the framework for analyzing the evolution of social communities developed by Goldberg et al.,[6] we tracked the evolution of CS researcher communities by searching for overlapping communities over consecutive time periods. We used the networks of authors represented as a bipartite graph in which each node representing a paper has edges to all nodes representing the paper's authors (see the table here). Figure 5 plots the number of communities that survived from one year to the next in the ACM and the IEEE datasets. The table lists average evolutionary chain length, average cluster size, average size of intersections of two to four consecutive clusters, and average relative density.[h] We found the recovered clusters had high average relative density of 0.8 for both datasets. The average length of the evolutionary chain was 4.5 years, while approximately two core researchers were associated with each cluster. This finding was consistent with the typical university team consisting of one or two stable faculty and three to five graduate students and postdocs joining and leaving continuously. Every four years or so, only a few stable researchers typically remained from an original research group.

Figure 5. Distribution of the length of evolutionary chains showing number of years a slowly evolving research community remains continuously active based on the ACM and IEEE datasets.



Evolution of research communities in terms of average size of a research group and number of years it was active based on the ACM and IEEE datasets.

Dataset	Average Value of	
ACM	Chain length	4.48
	Cluster size	6.1
	Intersection of two consecutive clusters	3.45
	Intersection of three consecutive clusters	2.51
	Intersection of four consecutive clusters	2.0
	Density	0.84
IEEE	Chain length	4.39
	Cluster size	5.53
	Intersection of two consecutive clusters	3.17
	Intersection of three consecutive clusters	2.36
	Intersection of four consecutive clusters	1.90
	Density	0.80

Conclusion

Most CS publications mention keyword algorithms, which is not surprising, and most abstracts mention one or more topics related to database, neural networks, and Internet. Our investigation also found the Web has become an attractive source of data and application testbeds, pulling in various researchers working on data mining, information retrieval, cloud computing, and networks. Most research related to the Internet has been done since 2000, even though the concept was introduced shortly after standardization of the TCP/IP protocol suite in early 1980s. Web pages evolved from simple text written in mark-up languages (such as HTML and XML) to the semantic Web, where ontologies are a key component for information retrieval by both humans and machines.

While overall trends provide a clear picture of the direction each topic is taking, the fraction of publications on each topic oscillates from year to year to the point the direction of change in one year consistently reversed in the subsequent year. The same pattern was reflected in the number of grants awarded for each topic each year. Since novelty is prized in publications and grant applications, authors tend to stress novel aspects of their work in abstracts and keywords, contributing to the observed pattern. We also found strong evidence of money preceding research; that is, if

h Average relative density is computed by dividing the combined weight of all edges with both endpoints in a cluster by the combined weight of all edges with at least one endpoint in the cluster.

a research topic bursts in terms of NSF grants first, it is likely to burst in publications within a few years. The opposite pattern is at least twice less frequent. Hence, we conclude while funding is not necessary in the initial growth in a CS research topic, it is essential for maintaining research momentum and researcher interest.

Most authors manage to publish at most once a year in a particular research field. Moreover, authors tend to publish in the same major research category for at most only a few years. Only a fraction of them continues to publish in the same field year after year for a long time. This agrees well with the model of an academic research team in which permanent faculty represent only a small fraction of the overall team of faculty, students, and postdocs, with the latter routinely changing topics after leaving a team. Moreover, a faculty member is often active in more than one area. Finally, since novelty is prized, authors tend to pursue new directions in their research, as reflected in an article's abstract and keywords, further contributing to the observed pattern.

During an NSF burstiness period, publication burstiness scores were more likely to increase than decrease, confirming that sustained NSF funding is essential for maintaining interest in a given topic.

Acknowledgment

The authors thank Francine Berman and James A. Hendler of Rensselaer Polytechnic Institute, George Cybenko of Dartmouth, and Jack Dongarra of the University of Tennessee, Knoxville, for discussions on the evolution of their research interests. The authors also thank Katie Bahran for help editing the article. The research was sponsored by the Army Research Laboratory and accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions in this article are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation here on.

References

1. ACM Digital Library; http://dl.acm.org
2. Agrawal, R., Ailamaki, A., Bernstein, P.A. et al. The Claremont report on database research. *Commun. ACM 52*, 6 (June 2009), 56–65.
3. Cohoon, J.M., Nigai, S., and Kaye, J. Gender and computing in conference papers. *Commun. ACM 54*, 8 (Aug. 2011), 72–80.
4. Edler, D. and M. Rosvall, M. Map Generator software package. MapEquation, Umeå, Sweden, 2010; http://www.mapequation.org
5. Fama, E.F. The behavior of stock-market prices. *Journal of Business 38*, 1 (Jan. 1965), 34–105.
6. Goldberg, M., Magdon-Ismail, M., Nambirajan, S., and Thompson, J. Tracking and predicting evolution of social communities. In *Proceedings of the Third IEEE International Conference on Social Computing* (Boston, Oct. 9–11). IEEE, 2011, 780–783.
7. Hall, M., Padua, D., and Pingali, K. Compiler research: The next 50 years. *Commun. ACM 52*, 2 (Feb. 2009), 60–67.
8. Hendler, J. and Berners-Lee, T. From the Semantic Web to social machines: A research challenge for AI on the World Wide Web. *Artificial Intelligence 174*, 2 (Feb. 2010), 156–161.
9. Hoonlor, A., Szymanski, B.K., Zaki, M.J., and Thompson, J. *An Evolution of Computer Science Research*. RPI Technical Report 12-01, Rensselaer Polytechnic Institute, Troy, NY, 2012; http://www.cs.rpi.edu/research/tr.html
10. Hoonlor, A., Szymanski, B.K., Zaki, M., and Chaoji, V. Document clustering with bursty information. *Computing and Informatics 31*, 6 (Dec. 2012), 1533–1555.
11. IEEE Xplore; http://ieeexplore.ieee.org/Xplore/
12. Lappas, T., Arai, B., Platakis, M., Kotsakos, D., and Gunopulos, D. On burstiness-aware search for document sequences. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France, June 28–July 1). ACM Press, New York, 2009, 477-486.
13. Moody, J. The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. *American Sociological Review 69*, 2 (Apr. 2004), 213–238.
14. National Science Foundation Award Search; http://www.nsf.gov/awardsearch/
15. Neilson, R.P. High-resolution climatic analysis and southwest biogeography. *Science 232*, 4746 (Apr. 1986), 27–34.
16. Porter, A.L. and Rafols, I. Is science becoming more interdisciplinary? Measuring and mapping six research fields over time. *Scientometrics 81*, 3 (Dec. 2009), 719–745.
17. Reacher, M.H., Shah, A., Livermore, D.M. et al. Bacteraemia and antibiotic resistance of its pathogens reported in England and Wales between 1990 and 1998: Trend analysis. *British Medical Journal 320*, 7229 (Jan. 2000), 213–216.
18. Rosvall, M and Bergstrom, C. Mapping change in large networks. *PLoS One 5*, 1 (Jan. 2010).
19. Rosvall, M. and Bergstrom, C. Maps of information flow reveal community structure in complex networks. *Proceedings of the National Academy of Sciences of the United States of America 105*, 4 (Jan. 2008), 1118–1123.
20. Ruzzo, W.L. and Tompa, M. A linear time algorithm for finding all maximal scoring subsequences. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology* (Heidelberg, Germany, Aug. 6–10). AAAI Press, Boston, 1999, 234–241.
21. Salehie, M. and Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems 4*, 2 (May 2009).
22. Thomson Reuters Web of Science; http://thomsonreuters.com/web-of-science/
23. Wikipedia, Computer Science; http://en.wikipedia.org/wiki/Computer_Science
24. Zaki, M.J. Sequences mining in categorical domains: Incorporating constraints. In *Proceedings of the Ninth ACM International Conference on Information and Knowledge Management* (Washington D.C., Nov. 6–11). ACM Press, New York, 2000, 422–429.

Apirak Hoonlor (apirak.hoo@mahidol.ac.th) is an instructor in the Faculty of Information and Communication Technology at Mahidol University, Bangkok, Thailand.

Boleslaw K. Szymanski (szymab@rpi.edu) is the Claire and Roland Schmitt Distinguished Professor of Computer Science and the Director of the Center for Network Science and Technology at Rensselaer Polytechnic Institute, Troy, NY.

Mohammed J. Zaki (zaki@cs.rpi.edu) is a professor in the Computer Science Department at Rensselaer Polytechnic Institute, Troy, NY.

DOI:10.1145/2494568

Quantum computer architecture holds the key to building commercially viable systems.

BY RODNEY VAN METER AND CLARE HORSMAN

A Blueprint for Building a Quantum Computer

SMALL-SCALE QUANTUM COMPUTING devices built on a variety of underlying physical implementations exist in the laboratory, where they have been evolving for over a decade, and have demonstrated the fundamental characteristics necessary for building systems. The challenge lies in extending these systems to be large enough, fast enough, and accurate enough to solve problems that are intractable for classical systems, such as the factoring of large numbers and the exact simulation of other quantum mechanical systems. The architecture of such a computer will be key to its performance. Structurally, when built, a "quantum computer" will in fact be a hybrid device, with quantum computing units serving as coprocessors to classical systems. The program, much control circuitry, and substantial pre- and post-processing functions will reside on the classical side of the system. The organization of the quantum system itself, the algorithmic workloads for which it is designed, its speed and capabilities in meeting those goals, its interfaces to the classical control logic, and the design of the classical control systems are all the responsibility of quantum computer architects.

In this article, we review the progress that has been made in developing architectures for full-scale quantum computers. We highlight the process of integrating the basic elements that have already been developed, and introduce the challenges that remain in delivering on the promise of quantum computing.

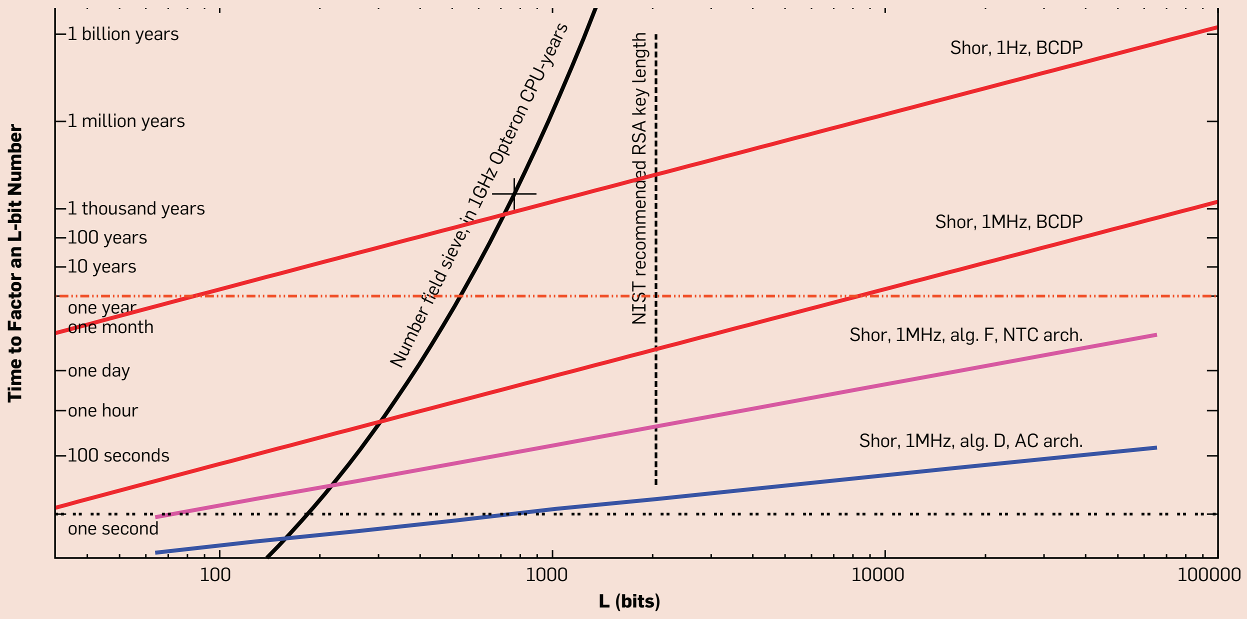
The most famous development to date in quantum algorithms is Shor's algorithm for factoring large numbers in polynomial time.[33] While the vernacular press often talks of factoring large numbers "in seconds" using a quantum computer, in reality it is not even possible to discuss the prospective performance of a system without knowing the physical and logical clock speed, the topology of the interconnect among the elements, the number of logical quantum bits (qubits) available in the system, and the details of the algorithmic implementation—in short, without specifying the architecture. Figure 1 illustrates the impact that architecture can have on the bottom-line viability of a quantum computer; here,

» key insights

- General-purpose quantum computers capable of efficiently solving difficult problems will be physically large, comprising millions or possibly billions of quantum bits in distributed systems.
- Quantum computer architecture matches available physical qubit technologies to applications.
- The driving force of an architecture is quantum error correction, guarding against loss of fragile quantum data.
- Even with quantum computers, constant factors matter as much as asymptotic computational class; researchers worldwide are working to bring error-corrected clock speeds up to operationally attractive levels.

Figure 1. Scaling the classical number field sieve (NFS) vs. Shor's quantum algorithm for factoring.[37]

The horizontal axis is the length of the number to be factored. The steep curve is NFS, with the marked point at $L = 768$ requiring 3,300 CPU-years. The vertical line at $L = 2048$ is NIST's 2007 recommendation for RSA key length for data intended to remain secure until 2030. The other lines are various combinations of quantum computer logical clock speed for a three-qubit operation known as a Toffoli gate (1Hz and 1MHz), method of implementing the arithmetic portion of Shor's algorithm (BCDP, D, and F), and quantum computer architecture (NTC and AC, with the primary difference being whether or not long-distance operations are supported). The assumed capacity of a machine in this graph is $2L^2$ logical qubits. This figure illustrates the difficulty of making pronouncements about the speed of quantum computers.



the architecture used can make the difference between an interesting proof-of-concept device and an immediate threat to all RSA encryption.

In developing a quantum computer architecture we have much to learn from classical computer architecture, but with a few important caveats. Foremost among these caveats is that the delicate nature of quantum information demands that memory elements be very active. Second, long wires or long-distance connections inside a quantum computer are either nonexistent, requiring nearest neighbor, cellular automaton-like transfer of data, or are at best poor quality, requiring much effort to transfer even a single qubit from place to place using quantum teleportation and error management techniques. Thus, the principles of classical computer architecture can be applied, but the answers arrived at likely will differ substantially from classical architectures.

Quantum computer architecture as a field remains in its infancy, but carries much promise for producing machines that vastly exceed current classical capabilities, for certain systems designed to solve certain problems. As we begin to design larger quantum computers, it must be recognized that large systems are not simply larger versions of small systems. The conceptual stack of subfields that must all contribute to a scalable, real-world machine is shown in Figure 2, divided into a set of layers. In this article, we discuss the elements of this structure in turn, all leading to the central core of quantum computer architecture.

At the bottom of the stack we have the technologies for storing individual qubits, and processing or transporting them to take part in a larger computation. How small groups of qubits will interconnect is the first problem quantum computer architecture must solve. Given the fragility of quantum data, how can many qubits be kept "alive" long enough to complete a complex computation? The solution to this problem, the field of *quantum error correction* (QEC), began with studies demonstrating that arbitrarily accurate computation is theoretically possible even with imperfect systems, but our concern here is the design of subsystems for executing QEC, which can be called the quantum computer microarchitecture.[26] Recent progress in experimentally demonstrated building blocks and the implementation of QEC are the first two topics addressed in this article.

At the top of the stack, machines will be designed for specific workloads, to run certain algorithms (such as factoring or simulation) that exist in computational complexity classes believed to be inaccessible to classical computers. Without these algorithms, there will be no economic incentive to build and deploy machines.

With context established at both the top and bottom of the stack, we present progress that has been made toward integrated architectures, and finish with a detailed example of the immense scale-up in size and slowdown in speed arising from the error correction needs of a full-scale, digital quantum computer. We note that the process of writing this article has been made substantially easier by the appearance in the last few years of excellent reviews on many architecture-relevant subfields.[1,3,4,9,13,19,28]

Qubit Technologies

At the lowest level of our Figure 2, we have the technological building blocks for the quantum computer. The first significant attempt to characterize the technology needed to build a computer came in the mid-1990s, when DiVincenzo listed criteria that a viable quantum computing technology must have: (1) a two-level physical system to function as a qubit;[a] (2) a means to initialize the qubits into a known state; (3) a universal set of gates between qubits; (4) measurement; and (5) long memory lifetime.[10] These criteria were later augmented with two communication criteria, the ability to convert between stationary and "flying" qubits, and the ability to transmit the latter between two locations.

In any qubit technology, the first criterion is the most vital: What is the state variable? Equivalent to the electrical charge in a classical computer, what aspect of the physical system encodes the basic "0" or "1" state? The initialization, gates, and measurement process then follow from this basic step. Many groups worldwide are currently working with a range of state variables, from the direction of quantum spin of an electron, atom, nucleus, or quantum dot, through the magnetic flux of a micron-scale current loop, to the state of a photon or photon group (its polarization, position or timing).

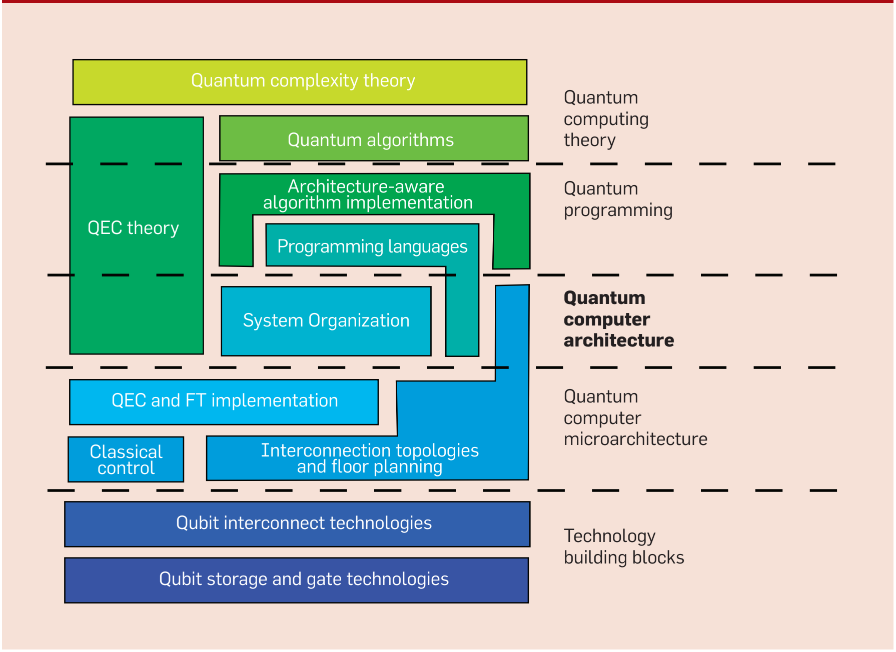
The accompanying table lists a selection of qubit technologies that have been demonstrated in the laboratory, with examples of the material and the final device given for each state variable.

Controlling any kind of physical system all the way down to the quantum level is difficult, interacting qubits with each other but not anything else is even harder, and control of systems for quantum computing over useful lifetimes is an immense experimental challenge. While experimental progress has been impressive in the last decade, moving away from one- and two-qubit demonstrations, we are still a very long way away from entangling, storing, and manipulating qubits on anything like the scale of classical computing and bits. Here, we are able to discuss only selected examples of the various technologies on offer; for more information, we recommend the recent survey by Ladd et al.[19]

Ion trap devices and optical systems currently lead in the number of qubits that can be held in a device,

a DiVincenzo's original criteria was written with qubits, on which we concentrate in this article, but tri-level qutrits or higher-dimension qudit data elements are also possible.

Figure 2. Quantum computer architecture among some sub-fields of quantum computation. QEC is quantum error correction; FT is fault tolerant.


Quantum complexity theory
Quantum algorithms
QEC theory
Architecture-aware algorithm implementation
Programming languages
System Organization
QEC and FT implementation
Classical control
Interconnection topologies and floor planning
Qubit interconnect technologies
Qubit storage and gate technologies
Quantum computing theory
Quantum programming
Quantum computer architecture
Quantum computer microarchitecture
Technology building blocks

A few of the many types of qubit technologies available. Different technologies rely on the same or different state variables to hold quantum data, implemented in different materials and devices. The equivalent technology for standard classical computing is given.

Type	Scaled CMOS (classical)	Ion trap	Quantum dot	Optical circuit	Gate-based superconducting circuit	Superconducting circuit (adiabatic computation)
State variable	Electrical charge	Ion spin	Electron spin, energy level, or position	Photon polarization, time, or position	Magnetic flux, charge, or current phase	Magnetic flux
Material	Doped silicon	Atoms in free-space electromagnetic field	Solid-state semi-conductor at cryogenic temperatures	Optical waveguides, for example, etched in silicon	Superconducting Josephson junction at cryogenic temperatures	Superconducting Josephson junction at cryogenic temperatures
Device gate	MOSFET	Laser- or vibrational-mediated interaction	Laser- or electrically-driven exchanges and rotations	Beam splitters and photon detectors	Electrically-driven exchanges and rotations	Electrically-controlled couplers
Maximum demonstrated variables	$> 10^9$ transistors per chip, $\approx 10^{16}$ per supercomputing system	14	3	8	2 full qubits + 2 special-purpose memories	8 coupled, 50 functional?

and controlled and entangled. Ions are trapped in a vacuum by electromagnetic potentials, and 14-qubit entanglement has been demonstrated (the largest entangled state in any form shown to date).[27] Complex bench-top linear optical circuits are capable of entangling eight-photon qubit states, and have been shown to perform nontrivial computation over short timescales.[40] Both of these approaches do not scale in those forms, but scalable approaches are also under development. Groups headed by Wineland, Monroe, Chuang, and others have demonstrated the necessary building blocks for ion traps.[21] Integrated nanophotonics (using photons as qubits) made on silicon chips provides the route to getting optics off of the lab bench and into easily scalable systems, and is making substantial progress in groups such as O'Brien's.[19]

Solid-state electronic devices for gate-based computation, while currently trailing in terms of size of entangled state demonstrated, hold out great promise for mass fabrication of qubits.[19] By trapping a single extra electron in a 3D block of semiconducting material, a quantum dot has a quantum spin relative to its surrounding that can hold a qubit of data. For flux qubits, the state variable is the quantum state of the magnetic flux generated by a micron-scale ring of current in a loop of superconducting wire (Figure 3).

In solid-state technologies, the experimental focus has been on improving the control of individual qubits rather than growing their numbers, but that focus has begun to shift. Jaw-Shen Tsai has noted that superconducting qubit memory lifetime has more than doubled every year since 1999, and has now reached the point where quantum error correction becomes effective.

Overall, the prospects are very good for systems consisting of tens of qubits to appear in multiple technologies over the next few years, allowing experimental confirmation of the lower reaches of the scaling behavior of quantum algorithms and the effectiveness of quantum error correction.

However, one factor that is often unappreciated when looking at these qubit technologies is the physical scale of the devices, particularly in comparison with classical digital technologies. Many people associate quantum effects with tiny objects, but most of these technologies use devices that are enormous compared to the transistors in modern computer chips. Transistors really are reaching down to atomic scales, with vendors having shipped chips fabricated with a 22-nanometer process at the end of 2011. In these chips, the smallest features will be only about 40 times the silicon crystal lattice cell size. In contrast, although the atoms themselves are of course tiny, ion traps are limited to inter-atom spacing of perhaps tens of microns for RF and optical control. Nanophotonic systems will require components tens of microns across, to accommodate the 1.5μm wavelength light that is desirable for telecommunications and silicon optics. Superconducting flux qubits require a current ring microns across. All of these technologies result in qubit devices that are macroscopic, or nearly so, with areal densities a million times less than computer chips. This fact will have enormous impact on large-scale architectures, as we will see.

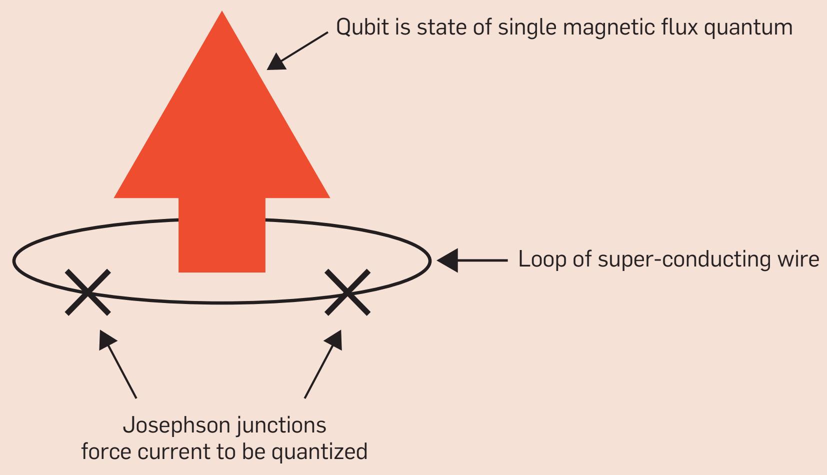
First steps in quantum architecture. At this lowest level, the job of quantum architecture is to determine how individual qubits or qubit blocks interconnect and communicate in order to process data. There are three main areas where experimental groups have begun to consider architectural implications in designing their systems.

Heterogeneity. Some researchers have been investigating technological heterogeneity by using combinations of electron spin, nuclear spin, magnetic flux, and photon polarization in a single system.[5] It is, however, equally important to consider structural heterogeneity, both in operational capability and in interconnect speed or fidelity. Martinis's group has recently demonstrated a chip with a functional distinction between two flux qubits and memory storage elements, leading them to refer to it as a quantum von Neumann architecture.[24] In many technologies, including some forms of quantum dots and Josephson junction qubits, measurement of a qubit requires an additional physical device, consuming die space and making layout of both classical and quantum components more complex.

Integration and classical control. Increasing the number of on-chip devices will require improving on-chip integration of control circuits and multiplexing of I/O pins to get away from multiple rack-mount units for controlling each individual qubit. Early efforts at considering the on-chip classical control overhead as systems grow include Oskin's design,[30] and

Figure 3. The Josephson junction superconducting flux qubit.

Here, tiny gaps in a superconducting wire (marked with *X*s), known as Josephson junctions, force the current in the loop to quantum tunnel across the gap, causing the current to be quantized. We can use a counterclockwise current (referred to as spin "up," as shown by the large arrow) to be a zero, and a clockwise current (spin "down") to be a one.


Qubit is state of single magnetic flux quantum
Loop of super-conducting wire
Josephson junctions
force current to be quantized

recent work by Levy et al. details the on-chip hardware's impact on error correction.[22] Kim uses his expertise in micro-mirrors to focus on the classical control systems for systems requiring switched optical control.[16,18] Controlling meso-scale systems—tens to low hundreds of physical qubits—will require substantial investment in systems engineering. Experimental laboratories likely will have to contract or hire professional staff with mixed-signal circuit design experience to do extensive control circuits, and for many technologies cryogenic circuits are required.

Interconnects. Even within individual chips, we are already seeing the beginning of designs with multiple types of interconnects. Integration levels will likely reach hundreds to low thousands of qubits in a single chip or subsystem, but reaching millions to billions of devices in a system will require interconnects that remain only on the drawing board. In some proposals, the inter-subsystem connections are physically similar to intra-subsystem connections, while in others they can be radically different. One ion trap proposal, for example, uses shared, quantum, vibrational modes for intra-node operations and optical connections between separate ion traps.[29]

Approaches to Quantum Information Processing

Data within a quantum computer is typically stored as *quantum bits* or *qubits*. Like a classical bit, a qubit has two states 0 and 1 but, unlike a classical bit, a qubit may be in a superposition of the two states, being in a certain sense in both 0 and 1 at the same time. A quantum gate operation that changes the state of this qubit can act on both values simultaneously. Each element in the superposition has a (complex) weight, say α for 0 and β for 1. When measuring a superposed state, only a single result (0 or 1) is returned, but the probability of measuring 0 is $|\alpha|^2$ and of measuring 1 is $|\beta|^2$. We cannot predict which outcome we will see, only its relative likelihood.

The power of superposition extends when we consider qubit registers: by analogy with a classical register, many qubits act together to store data as bit strings. In the quantum case, the register can be in a superposition of all possible register values. For example, a 3-qubit register can be in the superposed state of all eight values 000 to 111, all with different weights. In some such cases, when the superposition contains more than a single qubit, the qubits can be entangled with each other. The individual qubits no longer act independently, and exhibit much more strongly correlated behavior than is possible for classical systems. As with a single qubit, when quantum gates are performed on a register, operations are performed on all values simultaneously.

Extracting the relevant data is the difficult part of quantum computing. Only one element of a superposition can ever be measured, and we cannot choose which one it is. Algorithm designers aim to manipulate the weights so that, when the time comes to measure the result, the majority of the weight is given to a state that is a solution to the input problem.

Several different methods have been developed to use these fundamental elements of quantum computing. The most widely considered is circuit-based computation. Directly analogous with classical digital computation, data is stored in qubits and manipulated by the application of gate operations. In general, the first step of a circuit-based computation is to create an equal superposition of all register states. Gate operations between qubits then change the weights in the superposition, usually creating entanglement in the process.

A separate approach is *adiabatic quantum computation*. As with the circuit model, the output state is measured to give the final answer. In this case, however, the state is designed to be the low energy ground state of a quantum system in the quantum computer. The key to the computation is to adjust the coupling between quantum systems in the device to allow it to relax into that specific ground state.

Other approaches include *measurement-based quantum computation*, in which a large entangled state is reduced to the desired output state simply by carefully choosing how to measure the qubits, and *direct simulation*, in which the quantum states are designed to model a different physical system, rather than calculate a value numerically.

Error Correction

The heroic efforts of experimentalists have brought us to the point where approximately 10 qubits can be controlled and entangled. Getting to that stage has been a monumental task as the fragile quantum system must be isolated from the environment, and its state protected from drifting. What will happen, then, when we push to hundreds, thousands, or millions of qubits?

This brings us to the next level of Figure 2, the microarchitecture for quantum computers. If a quantum architecture were designed where gates of an algorithm were run directly on the types of individual physical qubits that we have been describing, it would not work. Both the accuracy of the gate operations and the degree of isolation from the environment required to perform robust computation of any significant size lie far outside the realm of feasibility. To make matters worse, quantum data is subject to a restriction known as the "no-cloning theorem" that means standard, classical, methods of controlling error cannot be implemented.[39] Quantum data cannot be "backed-up," or copied for simple repetition code processing to detect and correct for errors.

The possibility of performing quantum computation is saved, however, by quantum error correction. Some techniques are based on classical error correction and erasure correction, while others are based on uniquely quantum approaches.[9] In all cases, a number of physical qubits are combined to form one or more logical qubits. The number of physical qubits per logical qubit is determined by the quantum operation error rates, the physical memory lifetime, and the accuracy required of the algorithm, and can vary from tens to possibly thousands. A key property of a code is its threshold, the accuracy to which all physical qubit operations must perform in order for the code to work. Once enough physical qubits can be interacted to make interacting logical qubits, with all physical device operations accurate to below the threshold, then error correction will keep the quantum computer error-free for the runtime of the algorithm.

When assessing a classical error correcting code, an important aspect is the code rate, the ratio of delivered, corrected symbols to the number of raw symbols used. High rates are achieved in part by using very large blocks that encode many bits. Block-based codes have been explored in quantum codes, but suffer from the drawback that logical operations on logical qubits within the block are difficult to execute fault tolerantly. When selecting a quantum code, the rate is important, but the demands made on the implementing hardware and the ease of logical operations are critical.

Performing error correction is not a simple task; in fact, the vast majority of the processing power of a universal quantum computer will be used to correct errors in the quantum state. Application algorithms and data processing make their appearance only at a level well above the real-time, (physical) qubit-by-qubit work of error correction. An architecture for a universal quantum computer will therefore have as its primary goal the execution of specific types of error correction.

The earliest ideas for QEC naturally took advantage of classical error correction techniques. After solving the problems of measuring error syndromes without destroying the quantum state and computing on encoded states without inadvertently spreading errors (known in QC literature as fault tolerance, referring to runtime errors rather than mid-computation failure of hardware components), application of classical error correction became relatively straightforward.[9]

A promising form of error correction is surface code computation, which grew out of work by Kitaev and others on topological quantum computing. Raussendorf and collaborators created the 2D and 3D versions suitable for solid-state and photonic systems, respectively.[11,31] Fowler, Devitt, and others have extended the practicality of these results, including implementing the real-time error processing necessary to determine that the classical half of the machine is a tractable engineering problem.[8] The code rate of surface codes is poor, but their requirement only for nearest-neighbor connections will allow them to work at a higher physical error rate than other methods on some attractive hardware platforms.

Beyond digital quantum error correction for arbitrary states, other approaches can be used to (partially) isolate qubits from undesirable interactions. Decoherence-free subspaces encode a logical qubit in the phase difference of two physical qubits, suppressing the effect of certain types of noise. Techniques known as spin echo and dynamic decoupling similarly can be used to partially reverse the impact of systematic effects on memory, going with the error for a while and against it for a while, canceling out the effect. Purification—error detection for specific states—will be especially useful for communication, either system internal or external.

The implementation of error correction is perhaps the key near-term experimental goal. As experimental capabilities have grown, groups have begun competing to demonstrate quantum error correction in increasingly complete forms. Blatt's group performed multiple rounds of an error correction-like circuit that detects and corrects certain types of errors using certain simplifications.[32] Pan's group has recently shown an eight-photon entangled state related to the unit cell of 3D surface error correction.[40]

Microarchitectures for error correction. As in classical computer architecture, microarchitecture is the bridge between physical device capabilities and the architecture. Microarchitecture in the quantum case can be understood to be the level dedicated to efficient execution of quantum error management, while the system architecture is the organization of microarchitecture blocks into a complete system. There are several specifically quantum elements that must be considered for this microarchitecture.

Clock speed. The conversion factor from physical gate cycle to logical gate cycle has a strong, underappreciated impact on the performance of an algorithm. It depends on a number of architectural features, as well as the error correction code itself. For the ion trap-based system analyzed by Clark et al.,[6,26] a 10μsec physical gate results in a 1.6msec error correction cycle time using the basic form of Steane's error correcting code, which encodes one logical qubit in seven physical ones. The code will need to be applied in recursive fashion, resulting in growth of the physical system by an order of magnitude and an increase in the logical clock cycle to 260msec, not far below the topmost quantum line in Figure 1. This dramatic increase illustrates the effect of the base physical error rate on the architecture and performance, and will be discussed later.

Efficient ancilla factories. Most numeric quantum algorithms depend heavily on a three-qubit gate called a controlled-controlled NOT gate, or Toffoli gate. In most quantum error correction paradigms, direct execution of a Toffoli gate on encoded logical qubits is not possible. Instead, the Toffoli gate is performed using several operations, including one that consumes a specially prepared ancilla (temporary variable) state. The ancilla is created using distillation (a quantum error detection code), which takes noisy physical states and builds more accurate logical states. Creation of these states may dominate the workload of the machine, and recent work has assumed that 75%–90% of the machine is dedicated to their production. Isailovic et al. referred to this need as running quantum applications "at the speed of data," that is, producing the generic ancilla states rapidly enough that they are not the performance bottleneck.[14]

Balancing error management mechanisms. A functioning quantum computer almost certainly will not rely on a single type of error correction, but will incorporate different forms of error correction/detection/suppression at different levels. Different error management techniques have different strengths and weaknesses, and the combinatorial space for integrating multiple types is large.

Defects. A quantum architecture must also take into account that fabrication will inevitably be an imperfect process. Qubits may be declared defective because they fail to correctly hold the correct state variable carrier (for example, to trap a single electron), because memory lifetime is short or gate control imprecise, or because they fail to couple properly to other qubits. For gate-based, error-corrected systems, calculations show that a stringent definition of declaring a device to be functional pays for itself in reduced error correction demands.[36] A system's resilience to low yield is very microarchitecture-dependent. Alternatively, the digital quantum error correction itself can be adapted to tolerate loss.[34]

Workloads

So far we have looked at the lower two levels of Figure 2. Before investigating a complete quantum computer architecture, we need to consider the algorithms and programs that will be run on the physical hardware—the workload for the quantum computer. We therefore skip to the top of the stack: quantum programming and quantum algorithms. We are still in the time of Babbage, trying to figure out what Knuth, Lampson, and Torvalds will do with a quantum computer. It has been widely believed that Shor's factoring algorithm[33] and quantum simulation[3,4,20] will provide the two driving reasons to build machines. There are, however, a number of other useful and interesting quantum algorithms, seven of which are being investigated by teams involved in IARPA's Quantum Computer Science Program.[b] Bacon and van Dam[1] and Mosca[28] have published surveys covering quantum random walks, game theory, linear algebra, group theory, and more. Our understanding of how to design new quantum algorithms that asymptotically outperform classical ones continues to grow, though the number of people who can put the concepts into practice remains small.

Given the applications we have, how large a computer is needed to run them, and how should it be structured? Only a few quantum algorithms have been evaluated for suitability for actual implementation. Shor's algorithm is commonly used as a benchmark, both for its importance and clarity, and because the arithmetic and quantum Fourier transform on which it is founded are valuable building blocks for other algorithms.[7,12,26,36] Unfortunately, the size and speed of a machine needed to run the algorithm has been widely misunderstood. Architects have suggested a physical machine comprised of high millions to billions of qubits to factor a 2,048-bit number, a size that experimental physicists find staggering.[7,16,26,36]

In part because designs for a Shor machine have proven to be intimidatingly large, consensus is building that a Shor machine will not be the first commercially practical system, and interest in designing machines for quantum chemistry is growing. In a somewhat unexpected result, Brown's group has shown that certain quantum simulation algorithms expected to be computationally tractable on quantum computers are turning out to have dismayingly large resource requirements.[6] However, the field of quantum simulation is varied, and these simulators remain attractive; they are simply going to take more quantum computational resources (hence, more calendar years to develop and more dollars to deploy) than originally hoped.

A key element of the process of developing applications will be programming tools for quantum computers, and enough language designs were under way by 2005 to warrant a survey with a couple of hundred references.[13] In what are arguably the first examples of true "quantum programming," as distinct from "quantum algorithm design," shortly after the publication of Shor's factoring algorithm, Vedral et al. and Beckman et al. produced detailed descriptions of the circuits (sequences of gate operations) necessary for the modular exponentiation portion of the algorithm, which appeared as a single line in Shor's original paper.[2,38] The next step in implementation is to adapt such a description for execution on a particular machine, as in the block marked "Architecture-aware algorithm implementation" in Figure 2. Matching implementation choices to the strengths of the machine, including choosing adder circuits that match the application-level system interconnect, and trading off time and space, will be a collaboration between the programmer and the tools.[37] Maslov and others have studied efficient architecture-aware compilation,[25] an important element in the IARPA program. Compiler backends for specific experimental hardware configurations will soon be important, as will methods for debugging quantum programs in situ.

Quantum System Architectures

Finally we come to the central element in Figure 2. A complete system design will specify everything from the "business-level" requirements for an algorithm and machine capable of outperforming classical computers, through the details of the algorithm's implementation, the strength of error correction required and type of error management applied, the corresponding execution time of logical Toffoli gates (including the ancilla distillation discussed earlier), and the microarchitecture of individual areas of the system.

The DiVincenzo criteria are fundamental and necessary, but not sufficient to build a practical large-scale system. Considering instead the issues of quantum computer architecture results in a different focus, highlighting such mundane engineering criteria as being large enough and fast enough to be useful, and small enough and cheap enough to be built. Very loosely, meeting the DiVincenzo criteria can be viewed as the responsibility of experimental physicists, while the latter criteria are the responsibility of computer engineers.

Small-scale quantum architecture can be said to have begun with Lloyd's 1993 proposal for a molecular chain computer,[23] the first for a potentially buildable device. The word "scalable" attained a permanent position in the lexicon with Kielpinski et al.'s 2002 proposal for an ion trap that can shuffle and manipulate individual atoms,[17] an approach that continues to pay dividends. These ideas and many others for multiqubit devices, such as the quantum von Neumann approach or scalable ion traps with distinct operation and storage sites, sketch local areas of the system using the technological building blocks, but provide no direct guidance on how to organize large systems that meet the goal of solving one or more problems that are classically intractable.

When considering the macro architecture, certain aspects of the design become clear. Because all of memory is expected to be active, the system will probably not consist of separate CPU and memory banks connected via wide, shared buses. A more uniform array of microarchitecture error correction building blocks is the obvious approach, tempered by issues such as defective devices and the needs of the classical control subsystems. Each of these microarchitecture building blocks may utilize heterogeneous technology with an internal storage/computation distinction.[24] Individual chips or ion traps will not be large enough to execute some algorithms (notably Shor) at scale, likely forcing the adoption of a multicomputer structure.[15,29,36]

b http://www.iarpa.gov/solicitations.qcs.html. The IARP QCS was terminated in April 2013.

Large-scale designs are going to be difficult to create and evaluate without the appropriate tools. Further investment in automated tools for co-design of internally heterogeneous hardware and compilation of software is critical. One good example of this practice is Svore and Cross, working with Chuang, who have developed tool chains with round-trip engineering and error correction in mind.[35]

Architectural analyses exist for ion trap systems using Steane error correction, and multiple, distinct forms of nanophotonic and solid-state systems using the surface code.[7,16,26,36] We next take up one moderately complete architecture as an example.

An Architecture at Scale

We can use error management and application workloads to determine the broad outlines of a computer that could run a useful algorithm at full scale. The size of a quantum computer grows depending on the algorithm's demand for logical qubits, the quantum error correction scheme, the gate and memory error rate, and other factors such as the yield of functional qubits in the system. Overall, including space for various temporary variables and the ancilla state distillation, the scale-up factor from logical to physical qubits can reach half a million. As a specific example, the resource growth in one architecture[36] can be assigned approximately as follows:

- Establish a substantially post-classical goal of factoring an L= 2,048-bit number using Shor's algorithm, requiring
- $6L$ logical qubits to run a time-efficient form of the algorithm, growing by
- $8 \times$ to build "singular factories" for the state distillation process, allowing the algorithm to run at the speed of data,
- $1.33 \times$ to provide "wiring" room to move logical qubits around within the system,
- $10,000 \times$ to run the Raussendorf-Harrington-Goyal form of the surface code[31] with an error correcting code distance $d = 56$, and finally
- $4 \times$ to work around a yield of functional devices of 40%.

When will a quantum computer do science, rather than be science?

A singular factory is simply a region of the computer assigned by the programmer or compiler to the creation of the ancillae discussed here. The size, number, and position of the singular factories is dependent on the physical error rate and the size of the computation to be performed, and the type of interconnects available. The space for wiring is a surface code-dependent factor, not required when using other error correction methods, and is probably a substantial underestimate, though researchers are currently looking for compact compilations of programs on the surface code that will minimize this factor. The chosen code distance d is *strongly* dependent on the application algorithm itself and on the physical gate error rate. Shor's algorithm for $L = 2{,}048$ demands that, roughly speaking, we must be able to run 10^{15} logical operations with a high probability of correctly executing them all. This work was done assuming a physical error rate of 0.2%, which is not very far below the surface code threshold of 0.75%. These two factors determine the large distance of 56, and in the case of the surface code required resources grow as d^2, giving the high scale-up factor.[11] The final factor of four is strongly dependent on the details of the microarchitecture and the yield.

This results in a final system size of six billion physical qubits for the main quantum state itself, each of which must be independently controllable. In this particular architecture, this staggering number must be augmented with additional qubits for communications, on-chip optical switches, delay lines, and many external supporting lasers, optical switches, and measurement devices, all deployed in a large multicomputer configuration.

The performance of the system is determined by the error correction time and the complexity of executing the application gates on top of the error correction code. The surface code cycle time on this architecture for measuring all error syndromes is ~50μsec, far slower than the 100psec planned for physical gates. A Toffoli gate will require ~50msec—a factor of 1,000 from QEC cycle to logical gate, for this code distance. Demonstrating how system-level issues affect performance, the QEC cycle time is limited by contention for access to on-chip waveguides.

In part to address some of these architectural limitations, the QuDOS architecture was developed.[16] QuDOS, if built, would be 100 times faster, largely due to increased parallelism in measuring the error syndromes. Overall, coordinated changes of physical technology, error correction mechanism, and architecture may gain 3–4 orders of magnitude in performance, demonstrating the impact of the field of quantum computer architecture.

Conclusion

A principal lesson learned so far in research on quantum computer architectures is that systems capable of solving classically intractable problems will be large, although the search for the smallest commercially attractive machine continues. Device sizes will limit integration levels, affecting architecture, and determining logical clock speed requires making many design decisions but dramatically affects what can and cannot be effectively computed (as shown in Figure 1). Architectural problems cover a broad range, but have received only modest amounts of attention compared to near-term experimental hurdles, leaving much room for high-impact research that can help guide the focus of experimental work.

To sharpen the community focus on building systems, it seems to be time to begin demanding Moore's Law-like improvements in system capacity. Reviewers of papers and funding proposals should look for realistic estimates of logical Toffoli gate time, incorporating error correction, for some target logical fidelity. Even more ambitiously, we recommend requiring realistic estimates of application performance.

Developing a sense of community is critical. Creating a shared understanding including vocabulary, concepts, and important problems among the physics and CS theory, algorithm design, physics experiment, engineering, and architecture communities has proven to be difficult, and few journals or conferences currently provide good venues for such interdisciplinary endeavors, but we expect the number will grow.

Let us close with a question that provokes answers ranging from, "Already has," (in reference to direct quantum simulation of a specific quantum system[20]) to "Twenty years," to "Never,"—and all these from people actually working in the field:

When will the first paper appear in *Science* or *Nature* in which the point is the results of a quantum computation, rather than the machine itself? That is, when will a quantum computer do science, rather than be science?

Acknowledgments

This research is supported by the Cabinet Office, Government of Japan, and the Japan Society for the Promotion of Science (JSPS) through the Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program). We thank Simon Devitt, Dave Farber, Joe Touch, Michio Honda, Hajime Tazaki, Shigeya Suzuki, the referees and editors for careful reading of the manuscript. C

Supplemental Material
Additional information and references for this article are available under supplemental material in the ACM Digital Library DOI: 10.1145.249568.

References

1. Bacon, D. and van Dam, W. Recent progress in quantum algorithms. *Commun. ACM 53*, 2 (Feb. 2010), 84–93.
2. Beckman, D., Chari, A.N., Devabhaktuni, S. and Preskill, J. Efficient networks for quantum factoring. *Phys. Rev. A 54* (1996), 1034–1063; http://arXiv.org/quant-ph/9602016.
3. Brown, K.L., Munro, W.J. and Kendon, V.M. Using quantum computers for quantum simulation. *Entropy 12*, 11 (2010), 2268–2307.
4. Buluta, I. and Nori, F. Quantum Simulators. *Science 326*, 5949 (2009), 108–111.
5. Childress, L. et al. Coherent dynamics of coupled electron and nuclear spin qubits in diamond. *Science 314*, 5797 (2006), 281–285.
6. Clark, C.R., Metodi, T.S., Gasster, S.D. and Brown, K.R. Resource requirements for fault tolerant quantum simulation: The ground state of the transverse Ising model. *Phys. Rev. A 79*, 6 (June 2009).
7. Devitt, S.J., Fowler, A.G., Stephens, A.M., Greentree, A.D., Hollenberg, L.C.L., Munro, W.J. and Nemoto, K. Architectural design for a topological cluster state quantum computer. *New Journal of Physics 11* (2009).
8. Devitt, S.J., Fowler, A.G., Tilma, T., Munro, W.J. and Nemoto, K. Classical processing requirements for a topological quantum computing system. *International Journal of Quantum Information 8* (2010), 1–27.
9. Devitt, S.J., Nemoto, K. and Munro, W.J. Quantum error correction for beginners. *Reports on Progress in Physics 76*, 8 (Aug. 2013).
10. DiVincenzo, D. The physical implementation of quantum computation. *Fortschritte der Physik 48*, 9-11 (2000), 771–783.
11. Fowler, A., Mariantoni, M., Martinis, J. and Cleland, A. A primer on surface codes: Developing a machine language for a quantum computer. Arxiv preprint (2012); arXiv:1208.0928.
12. Fowler, A.G., Devitt, S.J. and Hollenberg, L.C. Implementation of Shor's algorithm on a linear nearest neighbor qubit array. *Quantum Information and Computation 4*, 4 (2004), 237.
13. Gay, S. Quantum programming languages: Survey and bibliography. *Bulletin of the European Association for Theoretical Computer Science* (June 2005).
14. Isailovic, N., Whitney, M., Patel, Y. and Kubiatowicz, J. Running a quantum circuit at the speed of data. *International Symposium on Computer Architecture*. IEEE (2008), 177–188.
15. Jiang, L., Taylor, J.M., Sørensen, A.S. and Lukin, M.D. Distributed quantum computation based on small quantum registers. *Phys. Rev. A 76* (Dec 2007).
16. Jones, N.C., Van Meter, R., Fowler, A.G., McMahon, P.L., Kim, J., Ladd, T.D. and Yamamoto, Y. Layered architecture for quantum computing. *Phys. Rev. 2*, 3 (July 2012), 031007.
17. Kielpinski, D., Monroe, C. and Wineland, D.J. Architecture for a large-scale ion-trap quantum computer. *Nature 417* (2002), 709–711.
18. Kim, J. and Kim, C. Integrated optical approach to trapped ion quantum computation. *Quantum Information and Computation 9*, 2 (2009).
19. Ladd, T., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C. and O'Brien, J. Quantum computers. *Nature 464* (Mar. 2010), 45–53.
20. Lanyon, B.P. Universal digital quantum simulation with trapped ions. *Science 334*, 6052 (2011), 57–61.
21. Leibrandt, D. et al. Demonstration of a scalable, multiplexed ion trap for quantum information processing. *Quantum Information and Computation 9*, 901 (2009).
22. Levy, J.E. et al. Implications of electronics constraints for solid-state quantum error correction and quantum circuit failure probability. *New Journal of Physics 13*, 8 (2011).
23. Lloyd, S. A potentially realizable quantum computer. *Science 261* (1993), 1569–1571.
24. Mariantoni, M. et al. Implementing the quantum von Neumann architecture with superconducting circuits. *Science 334*, 6052 (2011), 61–65.
25. Maslov, D., Falconer, S. and Mosca, M. Quantum Circuit Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27*, 4 (2008), 752–763.
26. Metodi, T.S., Thaker, D.D., Cross, A.W., Chong, F.T. and Chuang, I.L. A quantum logic array microarchitecture: Scalable quantum data movement and computation. In *Proceedings of the 2005 International Symposium on Microarchitecture* (2005).
27. Monz, T. et al. 14-qubit entanglement: Creation and coherence. *Phys. Rev. Lett 106*, 13 (Mar. 2011).
28. Mosca, M. Quantum algorithms (2008); Arxiv preprint arXiv:0808.0369.
29. Oi, D.K.L., Devitt, S.J. and Hollenberg, L.C.L. Scalable error correction in distributed ion trap computers. *Physical Review A 74*, 052313 (2006).
30. Oskin, M., Chong, F.T., Chuang, I.L., and Kubiatowicz, J. Building quantum wires: The long and short of it. In *Proceedings of the 30th Annual International Symposium on Computer Architecture* (June 2003), ACM, N.Y.
31. Raussendorf, R., Harrington, J. and Goyal, K. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics 9*, 199 (2007).
32. Schindler, P., Barreiro, J.T., Monz, T., Nebendahl, V., Nigg, D., Chwalla, M., Hennrich, M. and Blatt, R. Experimental repetitive quantum error correction. *Science 332*, 6033 (2011), 1059–1061.
33. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Alamitos, CA, 1994, 124–134.
34. Stace, T.M., Barrett, S.D. and Doherty, A.C. Thresholds for topological codes in the presence of loss. *Physical Review Letters 102*, 20 (2009).
35. Svore, K.M., Aho, A.V., Cross, A.W., Chuang, I. and Markov, I.L. A layered software architecture for quantum computing design tools. *IEEE Computer* (Jan 2006), 74–83.
36. Van Meter, R., Ladd, T.D., Fowler, A.G. and Yamamoto, Y. Distributed quantum computation architecture using semiconductor nanophotonics. *International Journal of Quantum Information 8* (2010), 295–323.
37. Van Meter III, R.D. Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm. Ph.D. thesis, Keio University, 2006; arXiv:quant-ph/0607065.
38. Vedral, V., Barenco, A. and Ekert, A. Quantum networks for elementary arithmetic operations. *Phys. Rev. A 54* (1996), 147–153; http://arXiv.org/quant-ph/9511018.
39. Wootters, W.K. and Zurek, W.H. A single quantum cannot be cloned. *Nature 299*, 802 (Oct. 1982).
40. Yao, X.-C. et al. Experimental demonstration of topological error correction. *Nature 482* (Feb. 2012), 489–494.

Rodney Van Meter (rdv@sfc.wide.ad.jp) is an associate professor in the Faculty of Environment and Information Studies at Keio University's Shonan Fujisawa Campus, Kanagawa Prefecture, Japan.

Clare Horsman (clare.horsman@gmail.com) was a project assistant professor at Keio University's SFC. She is currently a research assistant at the University of Oxford's Department of CS, Oxford, U.K.

research highlights

DOI:10.1145/2507771.2507782

Technical Perspective Can We Verify Cyber-Physical Systems?

By Rajeev Alur

OVER THE PAST few decades, computers have transformed from special-purpose and standalone number-crunching processors to networked devices interacting with the physical world. Realizing the full potential of such *cyber-physical systems* requires that advances in processing and communication technology are matched by advances in tools for designing such systems in a cost-effective manner. Indeed, none of us would be willing to drive an autonomous car, or use a pacemaker that can be remotely programmed by our physician, if we are not guaranteed safety under all conceivable conditions. The current practice in system design relies primarily on simulation that ensures the system works as intended only on a few scenarios selected by the design team. Formal verification, on the other hand, promises to catch bugs in corner cases by exploring the space of all possible executions of the system.

The verification problem is inherently computationally intractable, and yet researchers have made steady progress in developing methodology, algorithms, and tools for the verification of hardware and software systems. In particular, model checking—a technique for symbolically exploring the space of all reachable states of a system model—is considered a success story for *theory in practice*, due its adoption in checking correctness of multiprocessor designs and device driver code. In these applications, the model is purely discrete. To analyze a cyber-physical system such as a pacemaker, we need to consider the discrete software controller interacting with the physical world, which is typically modeled by differential equations. Developing effective symbolic verification technology for such mixed discrete-analog models—also called hybrid systems—has proved to be a challenging problem, attracting a lot of research both in formal methods and in control theory communities in recent years. The following paper by Althoff et al. reports a major milestone in this quest.

Since formal verification techniques have high computational costs, they need to be applied with a scalpel rather than with a hammer. The authors show great prudence in choosing their target. The design of *charge-pump phase-locked loops* is representative of both the technical challenges involved in the verification of hybrid systems, and of potential benefits of formal analysis. While the design has clearly specified requirements for stability and settling time, the discrete switching in the model implies that classical control-theoretic methods cannot ensure these analytically. The variations in initial conditions and parameter choices make it an ideal target for symbolic methods.

The experimental results reported in the following paper are truly significant since the authors demonstrate their method of symbolic simulation is competitive with the industry-standard simulation tools in terms of computational requirements.

Symbolic computation of reachable states of a hybrid system requires choosing a representation for sets of system states so the result of applying a single simulation step can be computed efficiently over this representation. Moreover, the complexity of the representation should grow manageably with the number of simulation steps. The technical strength of the paper is in the finely crafted combination of techniques the authors use to implement this symbolic computation. A key novelty of their method involves over-approximating the result of each simulation step by computing bounds on the range of switching times. This new method is critical in maintaining the simplicity of the symbolic representation across simulation steps, and allows the search to progress deep enough to establish stability.

The experimental results reported in the following paper are truly significant since the authors demonstrate their method of symbolic simulation is competitive with the industry-standard simulation tools in terms of computational requirements. This demonstration of cost-effectiveness is hopefully the key to convince commercial vendors such as Mathworks to invest seriously to explore the transition of verification technology for hybrid systems to industrial practice. C

Rajeev Alur (alur@cis.upenn.edu) is Zisman Family Professor of Computer and Information Science at University of Pennsylvania, Philadelphia, PA.

DOI:10.1145/2507771.2507783

Formal Verification of Phase-Locked Loops Using Reachability Analysis and Continuization

By Matthias Althoff, Akshay Rajhans, Bruce H. Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi

Abstract

We present a scalable and formal technique to verify locking time and stability for charge-pump phase-locked loops (PLLs). In contrast to the traditional simulation approach that only validates the PLL at a given operation condition, our proposed technique formally verified the PLL at all possible operation conditions. The dynamics of the PLL is described by a hybrid automaton, which incorporates the differential equations of the analog circuit elements as well as the switching logic of the digital circuit elements. Existing methods for computing reachable sets for hybrid automata cannot be used to verify the PLL model due to the large number of cycles required for locking. We develop a new method for computing effective overapproximations of the sets of states reached on each cycle by using uncertain parameters in a discrete-time model to represent the range of possible switching times, a technique we call *continuization*. Using this new method for reachability analysis, it is possible to verify locking specifications for a charge-pump PLL design for all possible initial states and parameter values in time comparable to the time required for a few simulation runs of the same behavioral model.

1. INTRODUCTION

In the standard design flow for analog mixed signal (AMS) circuits, the complete circuit is decomposed into its principal elements or *blocks*, which are first analyzed and designed using idealized low-order behavioral models. Detailed circuit-level designs are implemented only after the performance specifications have been verified at the block level over the required range of parameter variations and operating conditions. The goal is to create robust designs to avoid costly redesign cycles in the downstream process.

Because of the complexity of the mixed continuous and discrete (i.e., hybrid) AMS dynamics, there are no analytical techniques to verify a given design satisfies the circuit specifications, even for the simplified block-level behavioral models. Thus, numerical simulation has been the standard tool for evaluating the performance of behavioral models. Simulation is not completely satisfactory, however, because each simulation run represents the behavior for only one set of values for the initial states and parameters, so many simulations are required to assess the robustness of the design. Moreover, some specifications can be verified only after simulations have run for very long durations, and some specifications such as stability cannot be confirmed with absolute certainty because simulations cannot be run indefinitely.

This paper demonstrates an alternative to simulation based on formal methods. Formal methods offer an attractive alternative to simulation because they can verify that specifications for a circuit are satisfied for all possible behaviors over entire ranges of initial states and parameter values. This corresponds to an infinite number of simulation runs of unbounded duration. In their survey of the literature on formal verification for AMS designs, Zaki et al. categorize the methods into equivalence checking, automated state-space exploration, run-time verification, and proof-based methods.[11] Reachability analysis, the technique developed in this paper, is a form of automated state-space exploration.

The basic idea of reachability analysis is to use the dynamic equations for the circuit to propagate the trajectories of entire sets of states over time, rather than just a single state trajectory. The key issues are how to represent sets of states numerically and how to propagate these sets efficiently. Good techniques have been developed to represent and compute reachable sets for continuous dynamic systems (see e.g. Althoff[1] and Girard et al.[8]). All of these techniques are based on overapproximations, since the actual sets of reachable states are not convex in general. These overapproximations become less accurate as time progresses, however, and for hybrid dynamic systems the overapproximations become even less accurate and more time consuming to compute due to the need to compute overapproximations of intersections of reachable sets with the surfaces representing switching conditions.[2, 6] Therefore, current reachability analysis techniques for hybrid systems are effective when there are only a few discrete transitions in the time interval of interest.

To demonstrate the applicability of formal methods and reachability analysis to AMS circuits, we consider the verification of block-level behavioral models for a class of phase-locked loops (PLLs). PLLs are integrated circuits that produce high-frequency output signals that are synchronized to and in phase with low-frequency reference signals. Originally developed in the 1930's as a circuit for radio receivers, millions of PLLs are now used in virtually all digital

The original version of this paper was published in the *Proceedings of the International Conference on Computer Aided Design*, 2011, pp. 659–666.

communication systems, from satellites to mobile phones, as well as in many other applications such as clock generation for microprocessors. The charge-pump PLL is one of the popular PLL architectures.[7] It is an AMS circuit: the error signal driving the analog feedback is generated by digital logic.[7]

The primary requirements to be verified for a PLL are the circuit's *locking time* and *stability*. These specifications are illustrated in Figure 1. Locking time is a *transient specification*: the PLL state must reach the invariant region within a specified number of cycles. Stability is an *invariant specification*: from some set of initial states, the magnitude of the phase difference must remain within a given bound indefinitely. Both of these specifications must be achieved robustly, that is, from an arbitrary initial state and over a range of parameter values that reflect the target operating conditions (e.g., a given temperature range) as well as the inherent uncertainties that will arise from the detailed design and manufacturing processes. Verifying the behavioral model of a PLL using simulation is time consuming and ultimately inconclusive because: (i) locking can take a few thousand cycles, so very long simulation runs are required; (ii) each simulation run represents the behavior for only one set of values for the initial states and parameters, so many simulations are required to assess the robustness of the design; and (iii) invariance can only be inferred, but not guaranteed, because simulations cannot be run indefinitely.

We present a method for verifying both the transient and invariant specifications for a PLL over entire ranges of initial states and parameter values using reachable set computations that can be performed in the same amount of time currently required to simulate the circuit models for just a few selected points in the design space. Our approach relies on some new techniques tailored to the PLL problem because locking can require thousands of cycles, which implies there will be thousands of discrete transitions in the switching logic. Experiments with existing methods implemented in tools such as PHAVer[5] or SpaceEx[6] show that the overapproximations using existing methods become inaccurate so quickly that it is impossible to demonstrate that locking occurs, even for simple cases where locking can be demonstrated analytically.

The main technical contribution of this paper is a new method for computing accurate overapproximations of reachable sets for hybrid systems when there are a large number of discrete state transitions. This approach leverages previous results on computing reachable sets for linear systems with bounded uncertain parameters. Using the equations that govern the continuous dynamics of the PLL, we create a discrete-time model that generates tight overapproximations of the reachable sets at the beginning of each continuous-time cycle. Since the actual times at which the discrete transitions occur can vary, we introduce bounded uncertain parameters in the linear discrete-time model that account for the variations in the actual transition times. We call this process of mapping variations in time into parameter uncertainties *continuizaton*.[3] Finally, we show that satisfaction of the PLL specifications for the discrete-time model guarantees the specifications are satisfied at all points in time. The reachable sets for the discrete time model can be computed very fast, and the time reduced further by taking advantage of certain symmetries in the PLL dynamics. Our approach illustrates how the successful use of formal methods to solve real problems often requires extensions and insights that exploit the particular structure and features of the target application. It is an enabling technique that facilitates us to efficiently verify a PLL at all possible operation conditions.

We begin in the next section by showing how a class of charge-pump PLLs can be modeled at the behavioral level using hybrid automata with uncertain parameters. Section 3 presents a conversion of the continuous-time behavioral model to a discrete-time model, which provides the solution of the original model after each cycle. Variations in switching times of the PLL are abstracted away in Section 4 using the new concept of continuization. This makes it possible to abstract the hybrid dynamics of the PLL by a linear system with uncertain parameters. Using the model resulting from continuization, Section 5 presents the application of reachability analysis for formal verification of the PLL specifications, and Section 6 presents a comparison of the verification results using reachability to the classical simulation approach. The concluding section summarizes the contributions of this paper.

2. PLL BEHAVIORAL MODEL

We consider the dual path, type II, third-order charge-pump PLL shown in Figure 2, consisting of a reference signal generator (Ref), a voltage-controlled oscillator (VCO), a phase frequency detector (PFD), and charge pumps (CPs), along with RC circuits to implement a PI controller for the

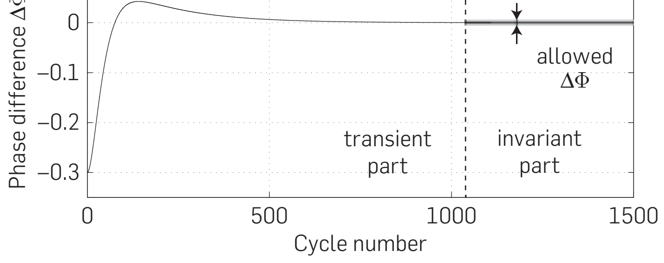


Figure 1. Transient (locking time) and invariant (stability) specifications for a PLL.

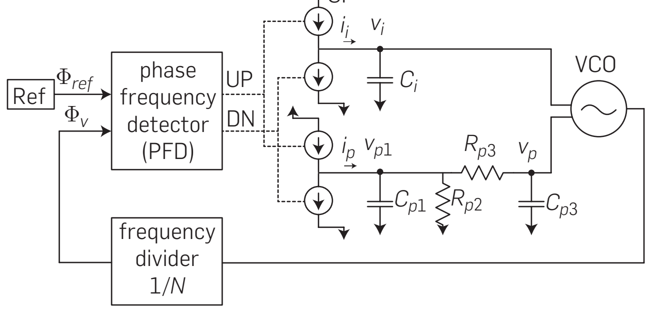


Figure 2. Dual-path charge-pump PLL.

feedback loop. The reference frequency generator produces a sinusoidal signal at a fixed low frequency (MHz), and the VCO generates a high-frequency signal (GHz). The desired output frequency of the VCO is determined by the reference frequency and the frequency divider ratio (i.e., N). The purpose of the PLL is to 'lock' the controlled frequency of the VCO so that its output has the same frequency (when divided by N) and phase as the reference signal.

Locking of the PLL is achieved by the PFD by comparing the phases of the reference signal and the VCO signal and setting the signals $\mathrm{UP} = 1$ if the reference signal leads, and $\mathrm{DN} = 1$ if it lags. These signals pump charge into or out of the capacitors, changing voltages v_p and v_i, which serve as proportional and integral (PI) control inputs to the VCO. For instance, if the reference signal leads, it means that the reference signal is faster than the VCO signal (when divided by N). In this case, UP is set to 1 and the "up" current will charge the capacitors so that the voltage values v_i and v_p increase. As a result, the VCO frequency increases in order to catch the reference signal. We do not consider adaptation of PLL parameters such as the frequency divider, resistor, or capacitor values.

As one can see from Figure 2, different components of the PLL system operate at different frequencies. For instance, the reference signal is at low frequency, while the VCO signal may be at extremely high frequency if the frequency divider ratio N is large. The large difference in frequency makes PLL simulation extremely challenging, since a traditional simulation tool must adopt a very small time step to numerically solve the PLL response in time domain. It, in turn, results in a very long simulation time.

The behavioral model of the charge-pump PLL is a hybrid automaton[4] with linear continuous dynamics and uncertain parameters. Appropriate bounds on the uncertain parameters can be determined by equivalence checking with detailed circuit models.[9, 10] These bounds should be chosen to assure that the behavioral model represents all possible behaviors of a detailed circuit model. If the more detailed model is at the transistor level, the approach is also able to catch issues at the transistor level. However, current equivalence checking techniques are typically semi-formal such that a complete enclosure cannot yet be guaranteed.

The continuous state vector in the behavioral model is $x = [v_i\, v_{p1}\, v_p\, \Phi_v\, \Phi_{ref}]^T$ with input vector $u = [i_i\, i_p]^T$ (see Figure 2). The dynamics are

$$\dot{x} = Ax + Bu + c, \tag{1}$$

with

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{C_{p1}}\left(\frac{1}{R_{p2}} + \frac{1}{R_{p3}}\right) & \frac{1}{C_{p1}R_{p3}} & 0 & 0 \\ 0 & \frac{1}{C_{p3}R_{p3}} & -\frac{1}{C_{p3}R_{p3}} & 0 & 0 \\ \frac{K_i}{N} & 0 & \frac{K_p}{N} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{1}{C_i} & 0 \\ 0 & \frac{1}{C_{p1}} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{2\pi}{N} f_0 \\ 2\pi f_{ref} \end{bmatrix},$$

where the resistor and capacitor values are given in Figure 2 and the values K_i, K_p, and f_0 determine the frequency of the VCO: $f_{VCO} = \frac{1}{2\pi}(K_i v_i + K_p v_p) + f_0$. Input values u vary depending on the signals leaving the PFD according to

$$u = \begin{cases} [I_i^{\mathrm{UP}} \quad I_p^{\mathrm{UP}}]^T, \text{ if } \mathrm{UP} = 1, \quad \mathrm{DN} = 0 \\ [I_i^{\mathrm{DN}} \quad I_p^{\mathrm{DN}}]^T, \text{ if } \mathrm{UP} = 0, \quad \mathrm{DN} = 1 \\ [I_i^{\mathrm{UP}} + I_i^{\mathrm{DN}} \quad I_p^{\mathrm{UP}} + I_p^{\mathrm{DN}}]^T, \text{ if } \mathrm{UP} = 1, \quad \mathrm{DN} = 1 \\ [0 \quad 0]^T, \text{ if } \mathrm{UP} = 0, \quad \mathrm{DN} = 0 \end{cases}$$

The output signals of the PFD are determined by threshold crossings of phase signals. The switching logic is described by the automaton shown in Figure 3, where the states are labeled as *up_active*, *dn_active*, *both_active*, and *both_off*.

Starting in *both_off*, the next discrete state of the hybrid automaton is *up_active* if the reference signal leads by first reaching $\Phi_{ref} = 2\pi$, and *dn_active* when $\Phi_v = 2\pi$ is reached first. As shown in Figure 4, in order to use the same phase crossings for the next cycle, the phase values are reset to $\Phi_{ref} := \Phi_{ref} - 2\pi$, $\Phi_v := \Phi_v - 2\pi$ upon continuing in *up_active* and *dn_active*. Once the lagging signal has a zero-crossing, the discrete state *both_active* is entered which models a time delay t_d for switching off both charge pumps. After the delay,

Figure 3. Hybrid automaton.

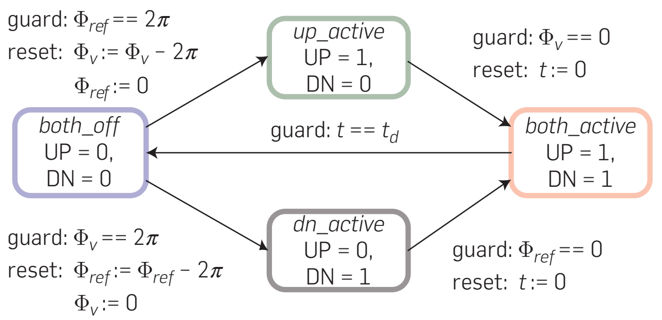
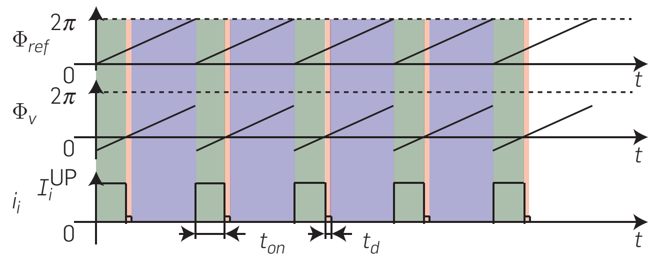


Figure 4. Typical charge pump activity.



the system is in *both_off* again, which completes one cycle. Locking is achieved when the phase difference reaches and remains within the *locked condition* given by the interval $[-0.1°, 0.1°]$.

3. TIME DISCRETIZATION

Given the hybrid automaton behavioral model of the PLL, we first derive a discrete-time linear model with bounded uncertain parameters based on the phase of the reference signal, assuming the reference signal leads the VCO signal, that is, for the discrete state sequence *up_active* → *both_active* → *both_off*. The time for a cycle of the reference signal is given by $t_{cycle} = 1/f_{ref}$. Since the continuous dynamics of the PLL is linear, we can take advantage of the superposition principle and obtain the initial state solution and the input solution separately. The initial state solution for one cycle is given by $x^h(t + t_{cycle}) = e^{At_{cycle}} x(t)$. The input solution for constant input u over the time interval $[0, r]$, where r is the time the charge pump is active, can be written using the Taylor series of e^{At} as

$$x^p(r) = \int_0^r e^{A(r-t)} dt\, u \in \underbrace{\Big(\sum_{i=0}^{\eta} \frac{1}{(i+1)!} A^i r^{i+1} \oplus \mathcal{E}^p(r) \Big)}_{=:\Gamma(r)} \otimes u, \qquad (2)$$

where $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$ is a Minkowski addition and $\mathcal{A} \otimes \mathcal{B} = \{ab | a \in \mathcal{A}, b \in \mathcal{B}\}$ a set-based multiplication. Note that sets can be sets of scalars, vectors, or matrices, and sets may also contain just a single (certain) element. The set multiplication sign is sometimes dropped when the context makes it clear that uncertain matrices are involved. The standard operator precedence rules apply. The set of remainders $\mathcal{E}^p(r)$ is overapproximated by an interval matrix, that is, a matrix with lower and upper bounds on each element, as presented in Althoff et al.[3] Since there is no input for the rest of the cycle, the input solution after one cycle is $x^p(t_{cycle}) \in e^{A(t_{cycle}-r)}\, \Gamma(r)u$. Let t_{on} denote the time the system is in location *up_active* and recall that t_d is the time it is in *both_active*. Also, let u denote the input in *up_active* and u_d denote the input in *both_active*. Finally, defining $x_k = x(k\, t_{cycle})$, the combination of the initial state solution and all input solutions can be written as

$$x_{k+1} \in \underbrace{e^{At_{cycle}} x_k}_{=x^h} \oplus \underbrace{\Gamma(t_{cycle})c}_{=:x^p_{const}} \oplus \underbrace{e^{A(t_{cycle}-t_{on})}\Gamma(t_{on})u}_{=:x^p_{up}} \oplus \underbrace{e^{A(t_{cycle}-t_{on}-t_d)}\Gamma(t_d)u_d}_{=:x^p_{both}}. \qquad (3)$$

The above formula is a discrete-time overapproximation of the continuous-time evolution after one cycle.

4. CONTINUIZATION

The model derived above computes the state of the system after one cycle when the switching time of the charge pumps is known. In this section, we develop a model that computes the range of state values that can occur at each cycle, for the entire range of possible switching times t_{on} (while t_d is a given constant). A closed form solution does not exist for t_{on}, which depends on the state of the system. Simulation techniques obtain t_{on} by detecting a zero-crossing which corresponds to crossing the guard condition $\Phi_v == 0$. Here we propose a more efficient method based on overapproximating the interval of possible values for t_{on}. Since t_{on} depends only on $\Phi_v = x_4$, it is sufficient to consider (see (1))

$$\dot{x}_4 = \frac{1}{N}(K_i x_1 + K_p x_3 + 2\pi f_0). \qquad (4)$$

We assume user-defined bounds $x_1 \in [\underline{\omega}_1, \overline{\omega}_1]$, $x_3 \in [\underline{\omega}_3, \overline{\omega}_3]$ which are monitored during the verification process. Violation of these bounds would require to restart the verification with larger intervals. Applying interval arithmetic to (4) results in the bound $\dot{x}_4 \in [\underline{\upsilon}, \overline{\upsilon}]$. We further extract the bound $[\underline{\delta}, \overline{\delta}]$ on x_4 from the reachable set at the beginning of each cycle. We obtain $t_{on} \in [\underline{t}_{on}, \overline{t}_{on}] = \{x_4 / \dot{x}_4 \,|\, x_4 \in [\underline{\delta}, \overline{\delta}], \dot{x}_4 \in [\underline{\upsilon}, \overline{\upsilon}]\}$ using the fact that the reference signal is leading ($\underline{\delta}, \overline{\delta} < 0$), resulting in

$$\underline{t}_{on} = |\delta| / \overline{\upsilon}, \quad \overline{t}_{on} = |\underline{\delta}| / \underline{\upsilon}. \qquad (5)$$

Using bounds on the switching times derived above, we use the concept of continuization to compute the set of reachable states resulting from uncertain switching times. To compute the reachable set under uncertain switching times (see Figure 5), we modify (3) and compute the solution successively, first of $\tilde{x}_k$ at times $t_k + \underline{t}_{on}$, then of $\tilde{x}_{k+1}$ at times t_{k+1}. This makes it possible to reset the uncertain switching time to values in the interval $[0, \Delta t_{on}]$, $\Delta t_{on} = \overline{t}_{on} - \underline{t}_{on}$ compared to $[\underline{t}_{on}, \overline{t}_{on}]$, which has computational benefits when evaluating Taylor series since higher order terms can be tightly bounded. The new equations are:

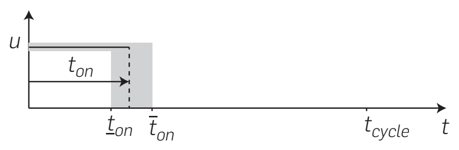
$$\begin{aligned} \tilde{x}_k &\in e^{A\underline{t}_{on}} x_k \oplus \Gamma(\underline{t}_{on})c \oplus \Gamma(\underline{t}_{on})u \\ x_{k+1} &\in e^{A(t_{cycle}-\underline{t}_{on})} \tilde{x}_k \oplus \Gamma(t_{cycle} - \underline{t}_{on})c \\ &\quad \oplus e^{A(t_{cycle}-\overline{t}_{on})} \underbrace{\Big\{ e^{A(\Delta t_{on}-\tilde{t})}\Gamma(\tilde{t}) \Big| \tilde{t} \in [0, \Delta t_{on}] \Big\}}_{=:\mathcal{G}(\Delta t_{on})} u \\ &\quad \oplus e^{A(t_{cycle}-\overline{t}_{on}-t_d)} \underbrace{\Big\{ e^{A\tilde{t}} \Big| \tilde{t} \in [0, \Delta t_{on}] \Big\}}_{=:\tilde{M}(\Delta t_{on})} \Gamma(t_d)u_d. \end{aligned} \qquad (6)$$

We drop the cycle index k for the following deviations for simplicity. It is possible to extract the time $\tilde{t}$ from the set $\mathcal{G}(\Delta t_{on})$, to obtain the relatively tight inclusion

$$\mathcal{G}(\Delta t_{on}) \subseteq \Big\{ \tilde{t}\,\mathcal{C}(\Delta t_{on}) \Big| \tilde{t} \in [0, \Delta t_{on}] \Big\},$$

where $\mathcal{C}(\Delta t_{on})$ is an interval matrix derived in Althoff et al.[3] Combining this result with $\tilde{t} \in \frac{|\tilde{x}_4|}{[\underline{\upsilon}, \overline{\upsilon}]}$ using the idea in (5) yields

Figure 5. Range of times $[\underline{t}_{on}, \overline{t}_{on}]$ when the charge pump is switched off. The mode *both_active* is not considered in this figure.



$$\mathcal{G}(\Delta t_{on}) \subseteq \mathcal{C}(\Delta t_{on})\frac{|\tilde{x}_4|}{[\underline{\upsilon},\overline{\upsilon}]}.$$

Now, the expression $e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{G}(\Delta t_{on})u$ in (6) is written in terms of $\tilde{x}$, independent of the current mode. Thereto, we use the fact that u only changes sign between modes based on the phase difference, which is taken care of by redefining the input $u := \mathrm{sgn}(x_4(t_k))(-u)$. We also consider that $u \in \mathcal{U}$, where $\mathcal{U}$ is an interval vector, such that

$$\begin{aligned} & e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{G}(\Delta t_{on})u \\ & \subseteq -e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{C}(\Delta t_{on})\frac{|\tilde{x}_4|}{[\underline{\upsilon},\overline{\upsilon}]}\mathrm{sgn}(\tilde{x}_4)\mathcal{U} \\ & = \underbrace{\left[\mathbf{0}\quad \mathbf{0}\quad \mathbf{0}\quad -\frac{1}{[\underline{\upsilon},\overline{\upsilon}]}e^{A(t_{cycle}-\overline{t}_{on})}\mathcal{C}(\Delta t_{on})\mathcal{U}\right]}_{:=\Theta(\Delta t_{on})} \otimes \tilde{x}, \end{aligned} \quad (7)$$

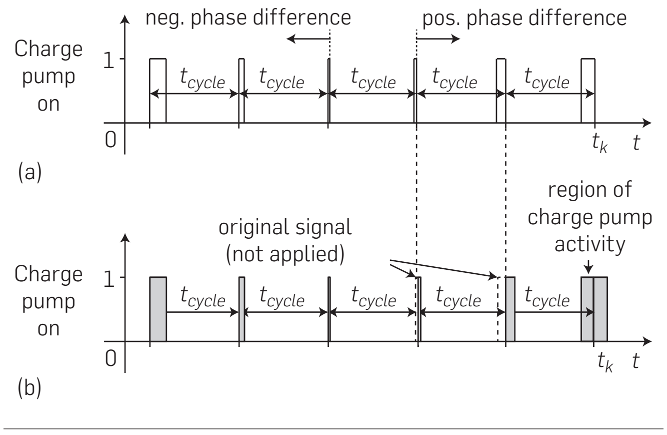
where $\mathbf{0}$ represents zero vectors of proper dimension and $\Theta(\Delta t_{on})$ is an interval matrix.

The result of (7) holds no matter if the phase difference is positive or negative as long as the time interval $[\underline{t}_{on}, \overline{t}_{on}]$ is correctly overapproximated. The time intervals computed in (5) are based on the cycle including *up_active*. If the cycle containing *dn_active* is also considered, the bounds are

$$\underline{t}_{on}=0, \quad \overline{t}_{on}=\max(|\underline{\delta}|/\overline{\upsilon}, |\underline{\delta}|/\underline{\upsilon}, |\overline{\delta}|/\underline{\upsilon}, |\overline{\delta}|/\overline{\upsilon}). \quad (8)$$

When computing the state bounds for a constant cycle time t_{cycle}, the input is applied in the interval $t_k+[0, t_{on}]$ for the cycle containing *up_active* and in the interval $t_k-[0, t_{on}]$ for the cycle containing *dn_active*. As shown in Algorithm 1 below, the different times are taken care of by adding the reachable set due to the input before and after t_k when both cycles are possible. The addition of the input solution for $t_k+[0, t_{on}]$ and $t_k-[0, t_{on}]$ results in an overapproximation since the input solution contains the origin, so that the previous sets are contained in the set after the addition. Further, it is sufficient to only keep the input applied at $t_k+[0, t_{on}]$ for subsequent computations, which is illustrated in Figure 6.

Figure 6. Consideration of inputs when the phase difference changes from negative to positive. (a) Signal of charge pump activity; (b) signal used for reachability analysis up to time t_k.



Thus, the error for adding the input solution for $t_k + [0, t_{on}]$ and $t_k - [0, t_{on}]$ does not accumulate. The procedure of only keeping the input applied at $t_k + [0, t_{on}]$ is realized by the auxiliary reachable set $\tilde{\mathcal{R}}_k$ in Algorithm 1. We skip the proof that this procedure is overapproximative due to space limitations.

5. REACHABILITY ANALYSIS

We now present how to compute the reachable set for a set of initial states and a sequence of cycles. The reachable sets are represented using zonotopes which have a maximum complexity of $\mathcal{O}(n^3)$ with respect to the system dimension n for the required operations. A zonotope is defined as

$$\mathcal{Z} = \left\{x \in \mathbb{R}^n \,|\, x = c + \sum_{i=1}^{p} \beta_i g^{(i)}, \quad -1 \le \beta_i \le 1\right\},$$

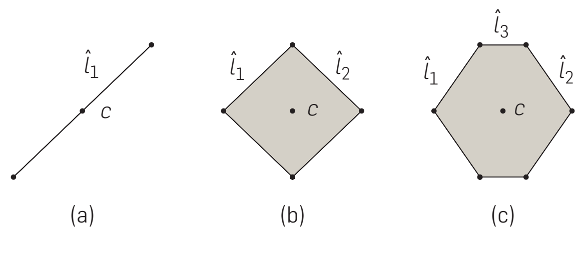
where $c \in \mathbb{R}^n$ is the zonotope center (to which a zonotope is centrally symmetric) and the $g^{(i)} \in \mathbb{R}^n$ are called generators. The order of a zonotope is defined as $o = \frac{p}{n}$. Figure 7 illustrates a zonotope being constructed step-by-step as the Minkowski sum of a finite set of line segments $\hat{l}_i = [-1, 1]g^{(i)}$. Operations on zonotopes and operations between sets of matrices and zonotopes are presented in Althoff.[1]

5.1. Transient analysis

The algorithm for the reachable set computation when the reference signal is initially leading is presented in Algorithm 1. An interesting property of the PLL is that the number of cycles required for locking is identical when the absolute value of the initial phase difference is equal and the corresponding initial voltages are symmetric with respect to the voltages in the completely locked state. We refer to this property as *symmetric locking time* which makes it sufficient to compute the reachable set only for the case when the reference signal is initially leading. For the symmetric locking, we additionally require that $I_i^{\mathrm{UP}} = -I_i^{\mathrm{DN}}$ and $I_p^{\mathrm{UP}} = -I_p^{\mathrm{DN}}$, which can be relaxed for reachability analysis by choosing the intervals for $I_i^{\mathrm{UP}} + I_i^{\mathrm{DN}}$ and $I_p^{\mathrm{UP}} + I_p^{\mathrm{DN}}$ large enough such that their center is 0. The proof for symmetric locking is omitted due to space limitations.

For simulation purposes, the values of the phases Φ_{ref} and Φ_v are needed to determine the time for turning the charge pumps on and off. In contrast, the discrete-time model for reachability analysis does not require the exact timing for switching the charge pump values; it is sufficient to keep only the phase difference $x_4 := \Phi_v - \Phi_{ref}$ as a state variable and remove x_5 for the reachability computations.

Figure 7. Construction of a zonotope by Minkowski addition of line segments. (a) $c \oplus \hat{l}_1$ (b) $c \oplus \hat{l}_1 \oplus \hat{l}_2$ (c) $c \oplus \hat{l}_1 \oplus \hat{l}_2 \oplus \hat{l}_3$



Algorithm 1 Reachable set computation when reference signal is leading at $t = 0$

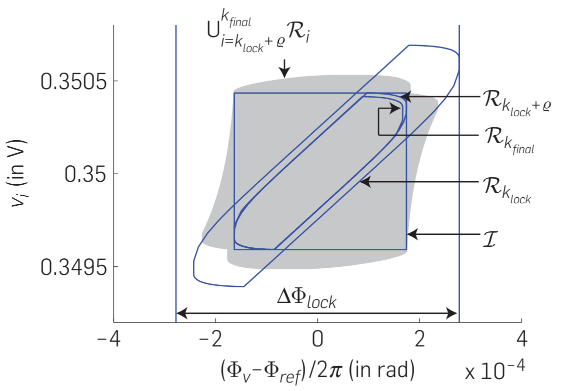
```
Input: Initial set $\mathcal{R}_0$, system matrix $A$, input set $\mathcal{U}$, input set
  $\tilde{\mathcal{U}}$ for $both\_active$
  parameters: $t_{cycle}$, $\Delta\Phi_{lock}$, $\underline{\upsilon}$, $\overline{\upsilon}$
Output: $\mathcal{R}_{k_{lock}}$
  $k = 0$;  $P = [0 \;\; 0 \;\; 0 \;\; 1]$;  $\tilde{\mathcal{R}}_0 = \mathcal{R}_0$
  while $|P\mathcal{R}_k| > \Delta\Phi_{lock}$ do
    $\underline{t}_{on} = \min(|P\mathcal{R}_k|)/\overline{\upsilon}$,  $\overline{t}_{on} = \max(|P\mathcal{R}_k|)/\underline{\upsilon}$
    Compute $\Gamma(t)$ for $t \in \{\underline{t}_{on}, t_d, (t_{cycle} - \underline{t}_{on})\}$; see (2)
    Compute $\Theta$ for $\Delta t_{on} = \overline{t}_{on} - \underline{t}_{on}$; see (7)
    Compute $\tilde{M}(\Delta t_{on})$; see (6)
    $\tilde{\tilde{\mathcal{R}}}_{k+1} = e^{A\underline{t}_{on}}\tilde{\mathcal{R}}_k \oplus \Gamma(\underline{t}_{on})c \oplus \Gamma(\underline{t}_{on})\mathcal{U}$
    $\tilde{\mathcal{R}}_{k+1} = \left(e^{A(t_{cycle} - \underline{t}_{on})} \oplus \Theta(\Delta t_{on})\right)\tilde{\tilde{\mathcal{R}}}_{k+1}$
        $\oplus \Gamma(t_{cycle} - \underline{t}_{on})c \oplus e^{A(t_{cycle} - \overline{t}_{on} - t_d)}\tilde{M}(\Delta t_{on})\Gamma(t_d)\tilde{\mathcal{U}}$
    $\mathcal{R}_{k+1} = \tilde{\mathcal{R}}_{k+1} \oplus \Theta(\Delta t_{on})\tilde{\mathcal{R}}_{k+1}$   (due to lagging)
    $k := k + 1$
  end while
  $k_{lock} = k - 1$
```

5.2. Invariant computation

Once the reachable set fulfills the locking condition $|P\mathcal{R}_k| \leq \Delta\Phi_{lock}$ (see Algorithm 1), it remains to check if this condition is fulfilled indefinitely. A straightforward procedure would be to check after each cycle if $\mathcal{R}_{k+1} \subseteq \mathcal{R}_k$, meaning that $\mathcal{R}_k$ is an invariant. Checking $\mathcal{R}_{k+1} \subseteq \mathcal{R}_k$ is computationally expensive. This is because zonotopes have to be represented by polytopes and the enclosure check for polytopes is computationally expensive.[1]

For this reason, we use the following alternative procedure illustrated in Figure 8. First, the reachable set computations are continued for ϱ extra cycles after a reachable set fulfills the locking condition in cycle k_{lock}, see Figure 8. Next, the reachable set $\mathcal{R}_{k_{lock}+\varrho}$ is overapproximated by an axis-aligned box denoted by $\mathcal{I}$. This leads to an overapproximation for the subsequent reachable sets, so ϱ should be chosen large enough such that all the subsequent sets fulfill the locking condition. Once a reachable set $\mathcal{R}_{k_{final}}$ represented by a zonotope is enclosed by $\mathcal{I}$ (which is computationally cheap to detect), one can conclude that the PLL is locked indefinitely.

Figure 8. Reachable sets of different stages of the invariant computation.



6. NUMERICAL RESULTS

We apply Algorithm 1 and the invariant computation to verify a 27 GHz PLL designed in 32 nm CMOS SOI technology. Note that the PLL was designed in a commercial process at an advanced technology node. Hence, it provides a practical example to demonstrate the efficacy of our proposed verification method. The parameters of the PLL and the reachable set computation are listed in Table 1. The PLL considered here employs a simple initialization circuitry that sets the integral and proportional path voltages to common-mode levels at power up and whenever the division ratio is changed. This reduces locking time and aids the formal verification by reducing the uncertainty on the initial node voltages. With the initialization, the initial range of node voltages are $v_i(0) \in [0.34, 0.36]$, and $v_{p1}(0)$, $v_p(0) \in [-0.01, 0.01]$. We normalize the phases to [0, 1], and we normalize the time to microseconds. The phase range of Φ_v is split into 5 subintervals $\Phi_v^i(0) \in -0.1\cdot[i, i-1]$, where $i = 1 \ldots 5$, and without loss of generality we assume $\Phi_{ref}(0) = 0$. Because of symmetry, all possible initial phase differences are considered. The number of Taylor terms chosen depends on the time horizon. For $\Gamma(t_{cycle} - \underline{t}_{on})$, 30 Taylor terms are used and 10 Taylor terms are used for all other computations. The aforementioned experiment setup allows us to formally verify the PLL with consideration of initial voltage and phase uncertainties. Note that these uncertainties cannot be efficiently incorporated into the traditional simulation approach, as a traditional simulation can only validate the PLL with a specific initial condition.

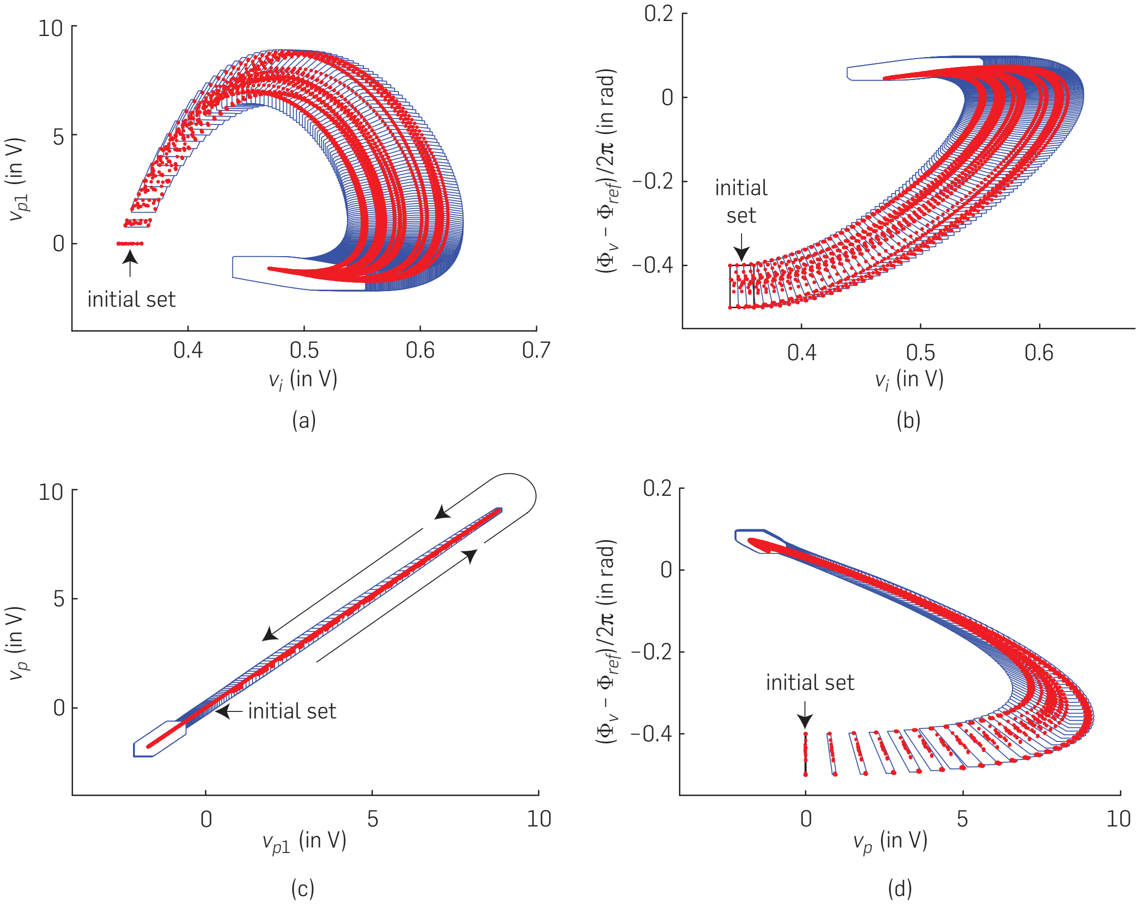
The reachable set starting with the initial phase difference $\Phi_v^1(0)$ is shown for the first 200 cycles in Figure 9 for projections onto four different pairs of state variables. The sets computed to prove locking are shown in Figure 8. In this example, the proposed verification algorithm is able to prove that independent of the initial condition, the PLL reliably locks to the reference signal. Note that the voltages in Figure 9 are as high as 10 [V] since charge pump saturation is not yet considered. It is possible to further extend our verification method to consider charge pump saturation by applying a nonlinear behavior model.

Table 2 shows the clock cycles it takes for the PLL to

Table 1. Parameters.

PLL model			Reachable set comp.	
Name	**Value**	**Unit**	**Name**	**Value**
f_{ref}	27	MHz	Max	
			zonotope	
f_0	26.93e3	MHz	Order o	100
N	1000	–	$\underline{\omega}_1$	0
K_i	200	MHz/V	$\overline{\omega}_1$	0.7
K_p	25	MHz/V	$\underline{\omega}_3$	−4
I_i	[9.9,10.1]e−6	A	$\overline{\omega}_3$	12
I_p	[495,505]e−6	A	ϱ	100
C_i	25e−12	F		
C_{p1}	6.3e−12	F		
C_{p3}	2e−12	F		
R_{p2}	50e3	Ohm		
R_{p3}	8e3	Ohm		
t_d	50e−12	s		

Figure 9. The blue regions show the reachable set of each cycle for the first 200 cycles. Simulation results of each cycle are plotted by red dots. (a) Projection onto v_i, v_{p1}; (b) projection onto v_i, $(\Phi_v - \Phi_{ref})/(2\pi)$; (c) projection onto v_{p1}, v_p; and (d) projection onto v_p, $(\Phi_v - \Phi_{ref})/(2\pi)$.



achieve locking for varying initial phase errors. The 1st and the 2nd columns show the results from reachability analysis. The 3rd column shows the maximum lock time obtained from 30 behavioral simulations with randomly varying initial phase errors and charge pump currents. We use the maximum lock time since the verification task is to check if the PLL always locks before a specified locking time, that is, we are investigating the worst-case behavior. Note that we are not providing any stochastic evaluation since this is not the focus of this work. Table 2 demonstrates that our reachability analysis efficiently provides an upper bound on the worst-case lock time in the presence of random phase error and charge pump current variations. On the other hand, it is important to note that the traditional approach based on Monte Carlo simulation cannot guarantee to find the "true" maximum lock time. Unless an infinite number of Monte Carlo runs are performed, the maximum lock time may not be captured by one of the Monte Carlo runs.

The computation times for the reachability analysis starting at different initial sets of phase differences are listed in Table 3. It can be seen that the results are obtained in less than a minute. The average computation time of the reachability analysis for a single cycle is around 27 [ms], which is only slightly longer than 24 [ms] required for a simulation of one cycle of the behavior model in MATLAB. All computations mentioned so far have been performed on an Intel i7 processor with 1.6 GHz and 6 GB memory. Simulating the behavioral model in VerilogA for a particular initial condition requires only 2 [ms] per cycle on an Intel Xeon CPU with 2.53 GHz, which is an order of magnitude faster than reachability analysis. However, reachability analysis is still

Table 2. Required cycles for locking.

	Reachability analysis		Simulation
$\Phi_v(0)$	Cycles to guarantee locking	Cycles to reach $\mathcal{I}$	(Max.) Cycles to reach $\mathcal{I}$
[−0.5,−0.4]	2039	1845	1271
[−0.4,−0.3]	1981	1787	1225
[−0.3,−0.2]	1908	1714	1173
[−0.2,−0.1]	1811	1616	1086
[−0.1,0]	1652	1457	994

Table 3. Computation times of the PLL. Computed number of cycles equals the left column of Table 2.

$\Phi_v(0) \in -0.1$	[5,4]	[4,3]	[3,2]	[2,1]	[1,0]
Comp. times (in s)	55.0	54.4	53.5	47.8	42.9

competitive if we consider that the VerilogA model needs to be simulated for thousands of Monte Carlo samples to capture random initial conditions and parameter variations.

7. CONCLUSION

This paper presented a method for verifying PLL locking using efficient reachability analysis. Efficient reachability computations are achieved using a discrete-time linear model with uncertain parameters and continuization to eliminate the complexity of switching. In contrast to applying a classical reachability approach, the intersection of guard sets can be dropped. As a consequence, the only operations on sets that remain, can be performed using zonotopes, which have a maximum complexity of $\mathcal{O}(n^3)$ with respect to the system dimension n. The verification of locking does not require any Lyapunov function to show convergence. For future work, we plan to consider saturations of charge pumps and varactor nonlinearities. We are also looking at other applications of continuization for hybrid systems where the transition time can be accurately overapproximated by a linear function of the state plus uncertainty. In addition to PLL, the proposed reachability analysis may be further extended to verify the circuit functionality and performance specifications of other AMS systems in time domain.

Acknowledgments

The authors acknowledge the support of the NSF Award CCF0926181 and the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity. C

References

1. Althoff, M. Reachability analysis and its application to the safety assessment of autonomous cars. Dissertation, Technische Universität München, 2010. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.
2. Althoff, M., Krogh, B.H. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control* (2012), 45–54.
3. Althoff, M., Rajhans, A., Krogh, B.H., Yaldiz, S., Li, X., Pileggi, L. Formal verification of phase-locked loops using reachability analysis and continuization. In *Proceedings of the International Conference on Computer Aided Design* (2011), 659–666.
4. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S. The algorithmic analysis of hybrid systems. *Theoretical Computer Science, 138* (1995), 3–34.
5. Frehse, G. PHAVer: Algorithmic verification of hybrid systems past HyTech. *Int. J. Software Tool. Tech. Tran. 10* (2008), 263–279.
6. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O. SpaceEx: Scalable verification of hybrid systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification* (2011), LNCS 6806, Springer, 379–395.
7. Garder, F.M. Phaselock Techniques, third edn, John Wiley, Hoboken, NJ, 2005.
8. Girard, A., Le Guernic, C., Maler, O. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control* (2006), LNCS 3927, Springer, 257–271.
9. Singh, A., Li, P. On behavioral model equivalence checking for large analog/mixed signal systems. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2010), 55–61.
10. Steinhorst, S., Hedrich, L. Advanced methods for equivalence checking of analog circuits with strong nonlinearities. *Formal Meth. Syst. Des. 36*, 2 (2010), 131–147.
11. Zaki, M.H., Tahar, S., Bois, G. Formal verification of analog and mixed signal designs: A survey. *Microelectron. J. 39*, 12 (2008), 1395–1404.

Matthias Althoff (matthias.althoff@tu-ilmenau.de), Ilmenau University of Technology, Ilmenau, Germany.

Akshay Rajhans, Bruce H. Krogh, Xin Li, and Larry Pileggi ({arajhans, krogh, xinli, and pileggi}@ece.cmu.edu), Carnegie Mellon University, Pittsburgh, PA.

Soner Yaldiz (soner.yaldiz@intel.com), Intel Corporation, Hillsboro, OR.

California State Polytechnic University, Pomona

Computer Science Department

Assistant Professor

http://www.csupomona.edu/~cs/

The Computer Science Department invites applications for a tenure-track position at the rank of **Assistant Professor** to begin Fall 2014. We are particularly interested in candidates with specialization in Software Engineering, Cloud Computing, Data Mining, or Computer Graphics and Animation. Cal Poly Pomona is 30 miles east of L.A. and is one of 23 campuses in the California State University. The department offers an ABET-accredited B.S. program and an M.S. program. Qualifications: Possess, or complete by August 31, 2014, a Ph.D. in Computer Science or closely related area. Demonstrate strong communication skills, commitment to actively engage in the teaching, research, and curricular development activities of the department at both undergraduate and graduate levels, and ability to work with a diverse student body and multicultural constituencies. Ability to teach a broad range of courses, and to articulate complex subject matter to students at all educational levels. First consideration will be given to completed applications received no later than November 15, 2013. Contact: Faculty Search Committee, Computer Science Department, Cal Poly Pomona, Pomona, CA 91768. Email: cs@csupomona.edu. Cal Poly Pomona is an Equal Opportunity, Affirmative Action Employer. Position announcement available at: http://academic.csupomona.edu/faculty/positions.aspx. Lawful authorization to work in US required for hiring.

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.

LASSONDE
SCHOOL OF ENGINEERING

YORK U UNIVERSITÉ UNIVERSITY

Tenure-Stream Faculty Openings: The Lassonde School of Engineering (LSE) at York University, Toronto, Canada seeks three (3) outstanding candidates in Electrical Engineering and Computer Science at the rank of Assistant or Associate Professor, to commence July 1, 2014, subject to budgetary approval. One Tier-2 Canada Research Chair with expertise in **Big Data Analytics and Scientific Visualization**, and emphasis in visual depictions of data from complex systems; scalable analytics; data-enabled methods; or/and applications in data driven design and domain sciences; 2 positions with expertise in the areas of (i) **Power Systems** (high power electronics and drives; smart grids; renewable energy resources; storage devices; intelligent control), and; (ii) **Medical Devices** (Disease diagnostic and treatment technologies; Bioengineering instrumentation for cellular and molecular analysis) or (iii) **Electronics** (Mechatronics; Nanoelectronics). The successful candidates must hold PhD degrees in the relevant discipline with outstanding track record of teaching, scholarly research, and professional achievement. These positions will play key roles in the establishment of the Renaissance Engineering initiative at the LSE, an ambitious $250 million dollar hiring 100 new faculty and staff, and expanding the student body by 1500. Applicants should submit a CV, statements of contributions in research, teaching, and curriculum development, three reference letters, and three sample research publications electronically at ***http://lassonde.yorku.ca/new-faculty/*** by October 31, 2013. For full position details, please visit ***http://www.yorku.ca/acadjobs***. York University is an affirmative action employer. The affirmative action Program can be found at ***http://www.yorku.ca/acadjobs*** or a copy can be obtained by calling the affirmative action office at 416-736.5713. All qualified candidates are encouraged to apply; however, Canadian citizens and permanent residents will be given priority.

Indiana University

School of Informatics and Computing

Faculty positions in Computer Science, Computer Security, and Data Science

The School of Informatics and Computing at Indiana University, Bloomington, invites applications for three positions beginning in Fall 2014, in the areas of **Computer Science** (all subareas), **Computer Security**, and **Data Science**. Data science applications are invited from candidates in all areas including data analytics and information extraction; data life cycle; data management, semantics, and infrastructure; data policy and security; data science foundations; and Big Data, including candidates with a record of achievement in industrial research.

Positions are open at all levels. Applicants should have a Ph.D.in the relevant area and a well-established record (senior level) or demonstrable potential for excellence in research and teaching (junior level).

The IU Bloomington School of Informatics and Computing is the first of its kind and among the largest in the country, with unsurpassed breadth. Its mission is to excel and lead in education, research, and outreach spanning and integrating the full breadth of computing and information technology. It includes the Dept. of Computer Science and Informatics and the Dept. of Information and Library Science, which comprise over 85 faculty and 850 graduate students. Its graduate degrees include Master's degrees in Computer Science, Security Informatics, Bioinformatics, Human Computer Interaction Design, Information Science, and Library Science, and Ph.D. degrees in Computer Science, Informatics, and Information Science. It is also known for its strong undergraduate programs.

Located in the wooded rolling hills of southern Indiana, Bloomington is a culturally thriving college town with a moderate cost of living and the amenities for an active lifestyle. IU is renowned for its top-ranked music school, high performance computing and networking facilities, and performing and fine arts.

Applicants should submit a CV, a statement of research and teaching, and names of 3 references (junior level) or 6 references (senior level) using the submissions link specific to the position: http://indiana.peopleadmin.com/postings/373 (**for computer science**), http://indiana.peopleadmin.com/postings/374 (**for computer security**), http://indiana.peopleadmin.com/postings/375 (**for data science**), or by mail (online applications preferred) to Faculty Search, Informatics and Computing, 919 E 10th St, Bloomington, IN 47408. Questions may be sent to faculty-search14@soic.indiana.edu. To assure full consideration completed applications in Computer Science must be received by November 15, 2013; applications in Computer Security and Data Science must be received by December 1, 2013.

Indiana University is an Equal Opportunity/Affirmative Action employer. Applications from women and minorities are strongly encouraged. IU Bloomington is vitally interested in the needs of Dual Career couples.

San Diego State University

College of Sciences
Department of Computer Science
Assistant Professor of Computer Science
(Full Time, Tenure Track)

The Department of Computer Science is seeking a tenure-track Assistant Professor. The applicant should have a doctorate degree in Computer Science, Computer Engineering, or a closely related field. The applicant should demonstrate the potential to develop a research program in networks and large data infrastructure and analysis, teach undergraduate- and graduate-level courses in Computer Science, and direct advanced graduate and undergraduate student research. Candidates are required to teach a variety of core and advanced courses in Computer Science such as assembly language, computer architecture, networks, and scientific databases.

Candidates are expected to develop a coherent funded program of research and remain active in the profession and in appropriate professional organizations. They must also be willing to engage in collaborative interdisciplinary research with faculty members from other departments. Research/Teaching/Employment experience in infrastructure development, analytics, and network security is desirable.

The Department offers Bachelor and Master of Science degree programs, certificate programs, and faculty may supervise PhD students through the Computational Sciences Doctoral program. The Department houses research in several computer science disciplines, and is committed to collaborative research experiences.

Salary is commensurate with experience.

To apply, send a letter of application, statement of teaching and research philosophy, curriculum vita, and three letters of recommendation to **Ms. Jensen, Administrative Coordinator, Department of Computer Science, San Diego State University, San Diego, CA 92182-7720**. Questions about the position may be directed to Dr. Beck, Department Chair, by email: beck@mail.sdsu.edu. Review of applications will commence in October, 2013 and continue until the position is filled.

SDSU is an equal opportunity employer and does not discriminate against persons on the basis of race, religion, national origin, sexual orientation, gender, gender identity and expression, marital status, age, disability, pregnancy, medical condition, or covered veteran status.

The person holding this position is considered a "mandated reporter" under the California Child Abuse and Neglect Reporting Act and is required to comply with the requirements set forth in CSU Executive Order 1083 as a condition of employment.

Texas State University

Department of Computer Science

Applications are invited for **two tenure-track faculty positions** in any computer science field to start September 1, 2014. The rank for either position can be **Assistant Professor or Associate Professor**, depending on qualifications. Consult the department recruiting page at http://www.cs.txstate.edu/recruitment/faculty_recruit.php for job duties, qualifications, application procedures, job position number, and information about the university and the department.

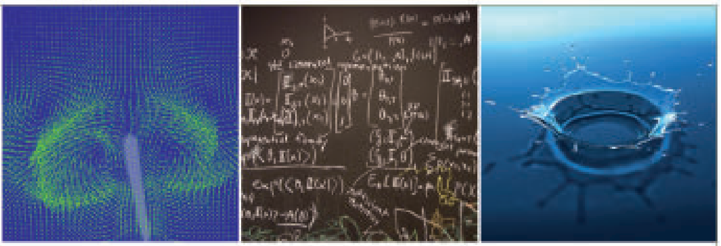
Texas State University (Texas State) will not discriminate against any person in employment or exclude any person from participating in or receiving the benefits of any of its activities or programs on any basis prohibited by law, including race, color, age, national origin, religion, sex, disability, veterans' status, or on the basis of sexual orientation. Texas State is committed to increasing the diversity of its faculty and senior administrative positions. Texas State is a member of The Texas State University System. Texas State is an EOE.

University of Chicago

Department of Computer Science
Assistant Professor

The Department of Computer Science at the **University of Chicago** invites applications from exceptionally qualified candidates in the areas of theory of computing, and systems for faculty positions at the rank of **Assistant Professor**.

Systems is a broad, synergistic collection of research areas spanning systems and networking, programming languages and software engineering, software and hardware architecture, data-intensive computing and databases, graphics and visualization, and systems biology. Particular areas of focus include formal definition, design, and implementation of programming languages, da-



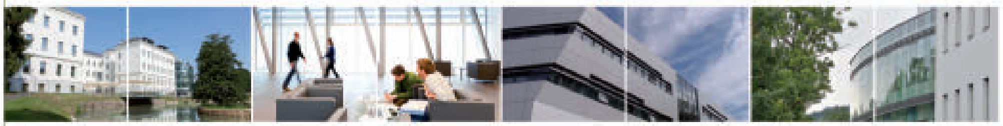
CALL FOR
Professors and Assistant Professors

IST Austria invites applications for **tenured** and **tenure-track leaders** of **independent research groups** in following fields: **Computer Science | Mathematics | Physics | Chemistry | Biology | Neuroscience | Earth Science | Interdisciplinary Areas**

The Institute is dedicated to basic research and graduate education in the natural and formal sciences. The successful candidates will receive a substantial annual research budget, are expected to apply for external research grants and to participate in the Graduate School.

Deadline for receiving Assistant Professor applications: November 15, 2013
Open call for Professor applications
Further information and online application: **www.ist.ac.at/professor-applications**

IST Austria values diversity and is committed to equality. Female researchers are encouraged to apply.



IST AUSTRIA
Institute of Science and Technology

ENGINEERING AT ILLINOIS

DRIVE YOUR VISION

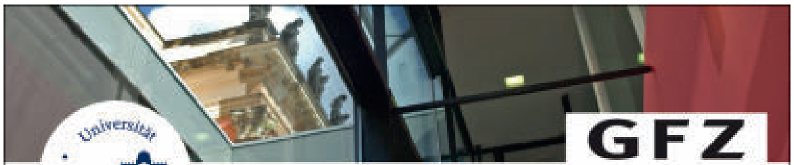
ENDOWED CHAIRS AND PROFESSORSHIPS IN BIG DATA

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.



Universität Potsdam

GFZ

Helmholtz Centre
POTSDAM

The **Faculty of Science of the University of Potsdam** and the **GFZ - German Research Centre for Geosciences** invites applications from outstanding individuals for a joint

W2-Professorship of Applied Computer Science with Applications in Geosciences

The position is a five-year temporary position with a tenure track option.

The professorship is based for the first five years at the University of Potsdam with subsequent leave of absence to the GFZ and simultaneous official secondment to the HPI.

Candidates with strong expertise in one or more of the following topics will be considered:

- Efficient algorithms and data structures for spatiotemporal data
- Scalable computation of massive spatiotemporal data
- Processing and management of large and heterogeneous spatial and spatiotemporal data
- Geoinformation services, geodata infrastructures and service oriented architectures
- Knowledge discovery from spatial and spatiotemporal data bases
- Information analytics for spatiotemporal data and visualization

His or her research areas should be explicitly related to geodata and geoinformation. The candidate will find a stimulating and multifaceted scientific environment within the German Research Center for Geosciences, the University of Potsdam, and the Hasso-Plattner-Institute. He or she will teach four hours a week at master level, two hours at the Hasso-Plattner Institute and two hours in other master programmes of the Faculty of Science.

The hiring process is subject to the requirements set forth in section 39 of the Brandenburg Higher Education Act for tenured professors only.

Send applications within four weeks after publication to Potsdam University, Office of the President, Am Neuen Palais 10, 14469 Potsdam, Germany or per E-Mail to president@uni-potsdam.de.

Further information you'll find here:

www.uni-potsdam.de
/verwaltung/dezernat3/stellen/

ta-intensive computing systems and algorithms, large scale distributed and collaborative systems, heterogeneous computer architectures, reliable computing systems, and self-tuning systems.

The University of Chicago has the highest standards for scholarship and faculty quality, and encourages collaboration across disciplines. We encourage strong connections with researchers across the campus in such areas as mathematics, natural language processing, bioinformatics, logic, molecular engineering, and machine learning, to mention just a few.

Applicants must have completed all requirements for the PhD except the dissertation at time of application, and must have completed all requirements for the PhD at time of appointment. The PhD should be in Computer Science or a related field such as Mathematics or Statistics.

The Department of Computer Science (cs.uchicago.edu) is the hub of a large, diverse computing community of two hundred researchers focused on advancing foundations of computing and driving its most advanced applications. Long distinguished in theoretical computer science and artificial intelligence, the Department is now building a strong Systems research group. This closely-knit community includes the Department of Statistics, Toyota Technological Institute, the Computation Institute, and Argonne's Mathematics and Computer Science Division.

The Chicago metropolitan area provides a diverse and exciting environment. The local economy is vigorous, with international stature in banking, trade, commerce, manufacturing, and transportation, while the cultural scene includes diverse cultures, vibrant theater, world-renowned symphony, opera, jazz, and blues. The University is located in Hyde Park, a Chicago neighborhood on the Lake Michigan shore just a few minutes from downtown.

All applicants must apply through the University's Academic Jobs website.

Please apply at the following sites:
1. Theory of computing, academiccareers.uchicago.edu/applicants/Central?quickFind=52933.
2. Systems, academiccareers.uchicago.edu/applicants/Central?quickFind=52953.

A cover letter, curriculum vitae including a list of publications, a statement describing past and current research accomplishments and outlining future research plans, and a description of teaching philosophy are required. Three reference letters are required, one of which must address the candidate's teaching ability to be considered as an applicant. The reference letters can be sent by mail to:

Chair, Department of Computer Science
The University of Chicago
1100 E. 58th Street, Ryerson Hall
Chicago, IL. 60637-1581

Or by email to: Recommend@mailman.cs.uchicago.edu (letters can be in pdf, postscript or Microsoft Word).

Candidates may also post a representative set of publications, as well as teaching evaluations, to this website.

To ensure fullest consideration of your application all materials, including supporting letters, should be received by January 15, 2014. However, screening will continue until all available positions are filled.

The University of Chicago is an Affirmative Action / Equal Opportunity Employer.

University of Notre Dame

Department of Computer Science and Engineering
Assistant or Associate Professor

The Department of Computer Science and Engineering at the **University of Notre Dame** invites applications for **Assistant or Associate Professor**. Excellent candidates in all areas will be considered.

The Department offers the PhD degree and undergraduate Computer Science and Computer Engineering degrees. Faculty are expected to excel in classroom teaching and to build and lead highly-visible research projects that attract substantial external funding.

The University of Notre Dame is a private, Catholic university with a doctoral research extensive

Carnegie classification, and consistently ranks in USNWR as a top-twenty national university. The South Bend area has a vibrant and diverse economy with affordable housing and excellent school systems, and is within easy driving distance of Chicago and Lake Michigan.

Applicants should send (pdf format preferred) a CV, statement of teaching and research interests, and contact information for three professional references to: cse-search-2014@nd.edu.

The University of Notre Dame is an Equal Opportunity, Affirmative Action Employer.

University of São Paulo

Institute of Mathematics and Statistics
Department of Computer Science
Tenure-Track Positions Assistant Professor

The Institute of Mathematics and Statistics of the University of São Paulo (IME-USP) invites applications for faculty positions at the Assistant Professor level. The Department is accepting applications in all areas of Computer Science.

We expect candidates with strong potential for research and teaching ability. The candidates should have a PhD in Computer Science or a related area. The selected candidate will be responsible for developing research and for teaching to the programs of the department (undergraduate and graduate courses). Deadlines and documents required for the application are specified at www.ime.usp.br/dcc/faculty_position The documents and selection interview may be either in Portuguese or English.

The University of São Paulo - USP is one of the most prestigious educational institutions in South America. It is the best ranked Brazilian university. The Department of Computer Science of the IME-USP is responsible for the BSc, MSc and PhD degrees in Computer Science, offering some of the most competitive courses in Brazil. The Department traditionally has open positions every year and potential candidates should contact us by email at any time.

More information: http://www.ime.usp.br/dcc
Contact: mac@ime.usp.br

Computer Science

Multiple senior faculty positions in computer science are available at Cornell's new Cornell NYC Tech campus in New York City. Faculty hired in these positions will be in the Department of Computer Science, which spans the Ithaca and New York City campuses, but their teaching and research will be based in New York City. We will consider only candidates at the Associate and Full Professor level, but will consider candidates from all areas of computer science and related fields.

Candidates must hold the equivalent of a Ph.D., must have demonstrated an ability to conduct outstanding research, and should also have a demonstrated interest in technology commercialization, entrepreneurship and social impact mission of the campus. In addition, experience in international programs and/or pre-college (K-12) education is of interest. This search may include Cornell faculty positions that are part of the Jacobs Technion-Cornell Innovation Institute. To ensure full consideration, applications should be received by December 1, 2013, but will be accepted until all positions are filled. Candidates should submit a curriculum vita, brief statements of research and teaching interests on-line at **https://academicjobsonline.org/ajo/jobs/2814**

Diversity and inclusion have been and continue to be a part of our heritage. Cornell University is a recognized EEO/AA employer and educator.

Computer Science

Multiple faculty positions are available at Cornell's Department of Computer Science, based in Ithaca, New York.

Candidates are invited to apply at all levels including tenured, tenure-track, or lecturer, and from all areas of computer science and related fields.

Tenured and tenure track faculty must hold the equivalent of a Ph.D.; applicants for the position must have demonstrated an ability to conduct outstanding research. Lecturers must hold the equivalent of a Master's degree, with a Ph.D. preferred.

To ensure full consideration, applications should be received by December 1, 2013, but will be accepted until all positions are filled.

Applicants should submit a curriculum vita, brief statements of research and teaching interests, and arrange to have at least three reference letters submitted at:
https://academicjobsonline.org/ajo/jobs/2813

Diversity and inclusion have been and continue to be a part of our heritage. Cornell University is a recognized EEO/AA employer and educator.

[CONTINUED FROM P. 112] were no sequential duplicates. And you're jumping around too much. A real random sequence would include some closer numbers."

She scratched her head. "But they use computers, right? And they can do random."

"Not really," said Morris. "They approximate with an algorithm. So Doug Hoffenmeyer checked his results to see if his subjects were just guessing. He had the idea of using my prototype quantum computer as the control. To be honest, he wanted to save money; he liked to keep his budget for his departmental outings with my wife."

She raised an eyebrow at Dan. "Quantum computer?" Wife?

Morris nodded. "It's my baby. There are hundreds of labs developing them, but mine's the first full prototype. A quantum computer uses quantum particles instead of binary bits to store information. It can work with infinitely long decimals. Transforms mathematical operations."

"Yeah, that's great, Piet," she said. "But why would it tell you to kill the guy?"

"This is where it gets messy. A week after he started using my numbers, Doug turned up complaining. Every value in the sequence was duplicated a week later in his tests. Naturally he thought somehow my computer was interfering with his. But we isolated them and still it happened. The numbers generated by the quantum computer predicted the values Doug's experiment produced a week later. Consistently. The damned computer was a precog."

A large drop of sweat had now formed on the tip of her nose. She swept it away, irritated. "You can't be serious? How?"

"We don't know for sure. Quantum theory suggests we live in a many-worlds universe, where all possible futures are played out. Maybe my quantum device was entangled with a future period. That would explain the fixed gap."

She spoke slowly. "But why the homicide?"

"Accident. I never intended to kill him. Doug was in my office talking about dropping his line of research, just concentrating on my computer.

He was no longer alive. He was deceased… would be deceased. In three days time.

I had to leave him a couple of hours for a faculty meeting. When I got back he was in shock. He'd used my computer to go online, to get the lottery results a week in advance. So then he wanted his password for the lottery website, to buy his ticket and make millions. The password was stored on the university notes database. Only it wouldn't let him in. Because he no longer worked there. He was no longer alive. He was deceased… would be deceased. In three days time. The machine was showing his future."

Minkowski shook his head. "So it said he would die. But why did you kill him, Doctor Morris?"

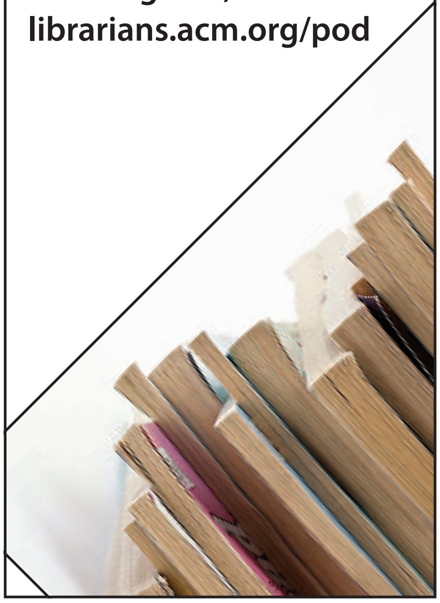
"We've heard the rumors." She watched him carefully. "Hoffenmeyer's affair with your wife. You killed him and thought you'd covered it up. Only we found out and now you've dreamed up this excuse."

"No," said Morris. "He was going to die anyway, but it was just an accident that I killed him. He ran out in front of my car. I couldn't stop in time."

Her head was thumping. Could it really be coincidental? Had the computer predicted the future or influenced it by prompting Morris to take revenge and pretend it was an accident? She stared at the still blank form in front of her. Her report could wait 'til morning.

Some things were better left in the future. C

Brian Clegg was a senior manager in the IT department of British Airways and is now a popular science author. His most recent book on randomness and probability in life and the universe *Dice World*, Icon Books, London, has been featured in a variety of publications.



From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2507771.2507784

Brian Clegg

Future Tense
Quantum Precog

Spared a horrible death, it was murder just the same.

SHE MOPPED THE beads of sweat from her brow. It was 3 A.M. but still 82 degrees in the interrogation room. Freakin' global warming. Her partner, Dan Minkowski, took over the questioning.

"Okay, Doctor Morris, you say the computer told you to kill your friend. Have you ever been diagnosed with schizophrenia?"

The accused slumped in his seat. He looked more like a singer in an 1980s punk band than a computer scientist. "No… look, I want to go back to the beginning."

She got in before Minkowski. This needed subtlety. "That's a good idea, Doctor Morris. Can I call you Piet?"

Morris nodded. "Fine, whatever. This would never have happened if Doug hadn't misused my prototype. He's… was a psychologist. Studying precognition."

"He'd used my computer to go online, to get the lottery results a week in advance."

Precognition. Working in Cambridge she'd got used to the odd preoccupations of Harvard and M.I.T. "Isn't that telling the future?"

"Kind of. Doug tested the ability of subjects to predict random events. The difficulty is assessing the statistical significance…"

"Okay, I get it," she interrupted. These academics couldn't resist lecturing.

"No, you need to know this. The experiments measure small deviations from randomness. Say you've got a computer program showing two closed curtains. It randomly chooses one or the other to open. The subjects guess ahead of time. Say they get a 52% hit rate. In the long run, that's significant. But it's easy to screw up the random stream."

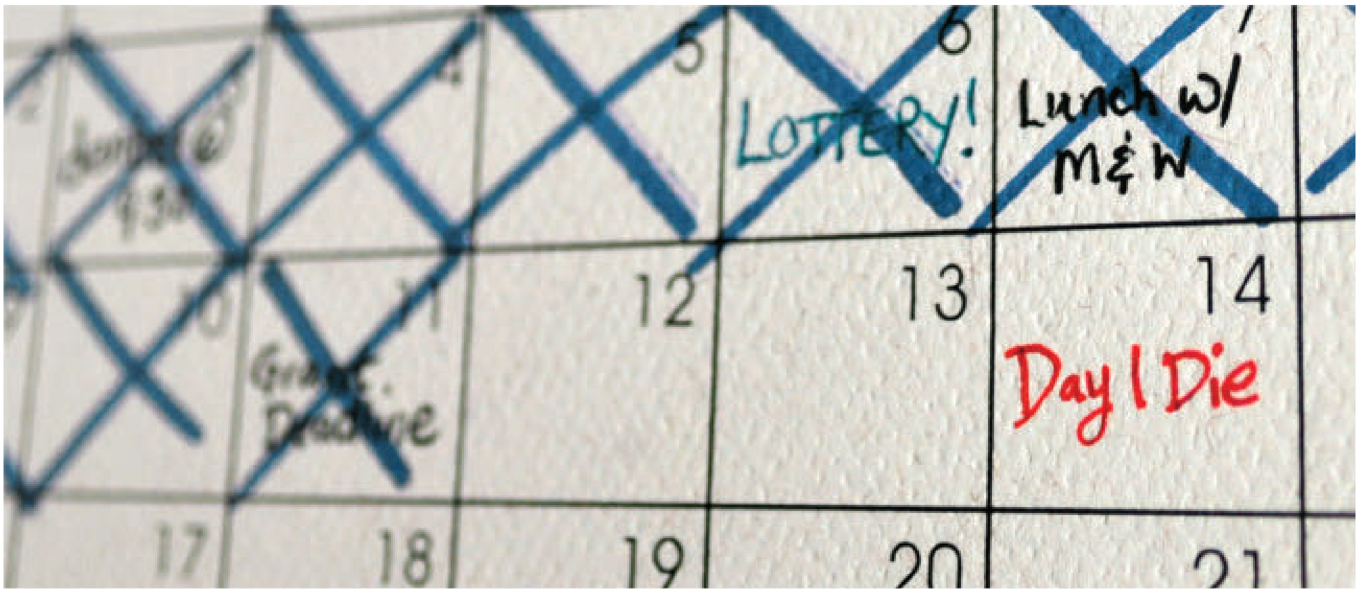
"Stream?" Minkowski had his baffled face on.

"Stream of numbers," said Morris. "Give me a dozen random numbers between 1 and 10."

"You want…?"

"Humor him," she cut in, not wanting Morris to clam up. "Okay, Piet, um 1, 7, 2, 4, 9, 5, 8, 1, 10, 7, 3, 5?"

"That's terrible." Morris shook his head. "There [CONTINUED ON P. 111]



SENSE THE TRANSFORMATION

REGISTER NOW TO SAVE

SIGGRAPH Asia offers registration categories to meet your needs and budget. Grab the opportunity to meet, network with, and learn from the industry's best talents.

Enjoy maximum savings when you register online with this code **SIGA1300286** before **11 November, 23:59 Hong Kong time to enjoy up to 20% off!**

Visit **sa2013.siggraph.org/registration-travel** for more details about registration.

SIGGRAPH ASIA 2013 HONG KONG

CONFERENCE 19 NOV - 22 NOV
EXHIBITION 20 NOV - 22 NOV

HONG KONG CONVENTION AND EXHIBITION CENTRE

SA2013.SIGGRAPH.ORG

LEAD SPONSOR CREATEHK

SPONSORED BY acm