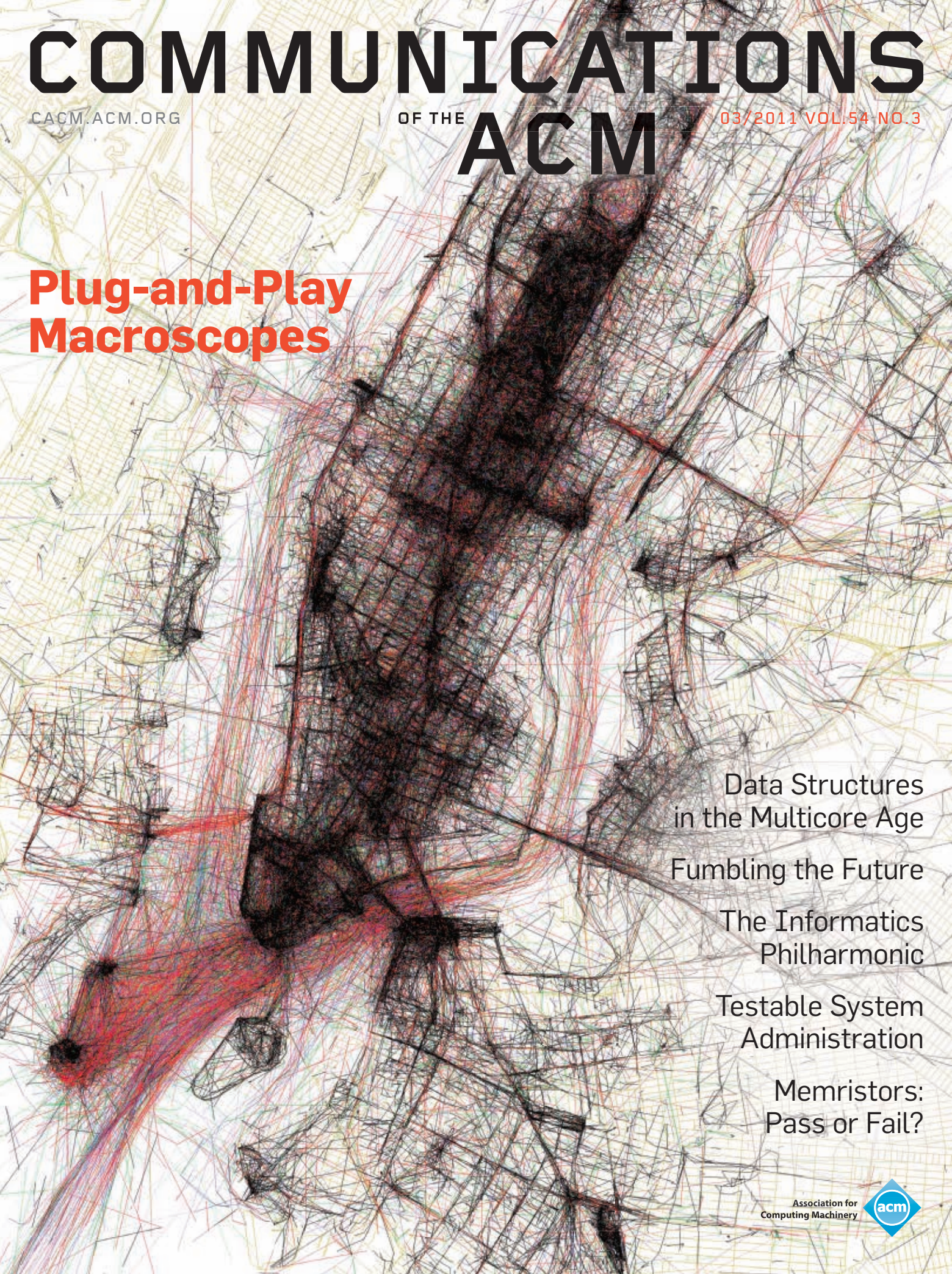
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

03/2011 VOL.54 NO.3

Plug-and-Play Macroscopes



Data Structures in the Multicore Age

Fumbling the Future

The Informatics Philharmonic

Testable System Administration

Memristors: Pass or Fail?

Association for Computing Machinery

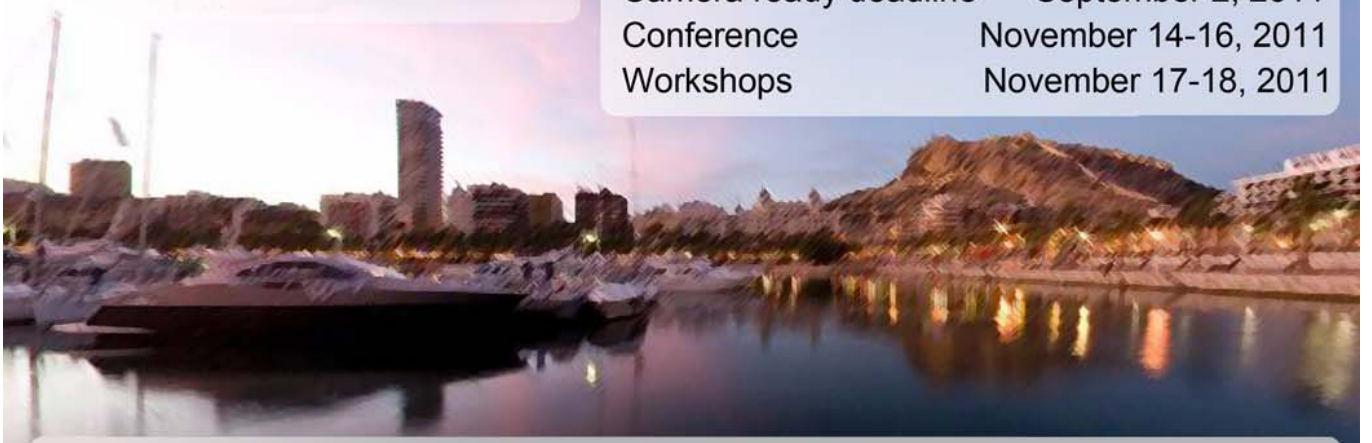
The International Conference on **Multimodal Interaction**

November 14-18, 2011
Alicante, Spain

ICMI is the premier international forum for multidisciplinary research on multimodal human-human and human-computer interaction, interfaces, and system development

Important dates

Workshops proposal	March 1, 2011
Paper and demo submission	May 13, 2011
Author notification	August 5, 2011
Camera ready deadline	September 2, 2011
Conference	November 14-16, 2011
Workshops	November 17-18, 2011



TOPICS OF INTEREST (but are not limited to):

* **Multimodal Interaction processing:**
Machine learning, pattern recognition and signal processing approaches for analysis and modeling of interaction, adaptation and multimodal input fusion and output generation, addressing any combinations of: vision, gaze, audio, speech, gestures, e-pen, haptic & tangible, bio-signals, such as brain and EMG, etc.

* **Interactive systems and applications:**
Mobile and ubiquitous systems, automotive and navigation, human-robot interaction, virtual and augmented reality, education, authoring, entertainment, gaming, telepresence, assistive systems, universal access, healthcare, biometry, intelligent environments, meeting analysis and meeting spaces, indexing, retrieval and summarization, etc.

* **Multimodal Interfaces:**
Design issues, principles and best practices & authoring techniques for human-machine interfaces using any combinations of input and/or output multiple modalities.

* **Modeling human communication patterns, affect and cognition:**
Multimodal human-human and human-machine communication; verbal and nonverbal interaction; discurse and dialogue modeling, multimodal social signal processing.

* **Data, evaluation and standards for Multimodal Interactive systems:**
Architectures; assessment techniques and methodologies; corpora; annotation and browsing of multimodal interactive data; standards for multimodal interfaces.

For more information, please visit the ICMI2011 website at:
http://www.acm.org/icmi/2011/

UNIVERSIDAD POLITECNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
Department of Software and Computing Systems
MINISTERIO DE CIENCIA E INNOVACION

COMMUNICATIONS OF THE ACM

Departments

Last Byte

Association for Computing Machinery
Advancing Computing as a Science & Profession

News

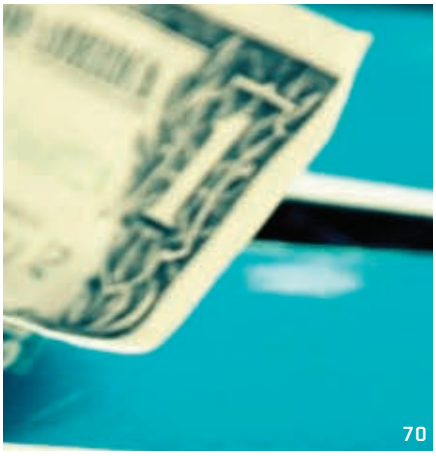
Viewpoints

03/2011
VOL. 54 NO. 03

Practice

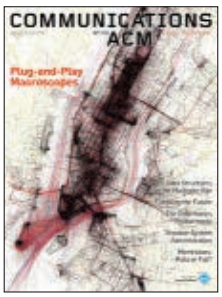
Articles' development led by acmqueue
queue.acm.org

Contributed Articles



Review Articles

Research Highlights



About the Cover: Artist/photographer Eric Fischer created the Geotaggers' World Atlas by collected geographical data from Flickr photos, revealing where people take pictures in major cities around the world. Each city was ranked on the density of photographs taken around its center; our cover image of New York City topped the list.

ILLUSTRATION BY ALEX WILLIAMSON. PHOTOGRAPH BY COLLIN PARKER

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) 626-0654.

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.

DOI:10.1145/1897852.1897853

Moshe Y. Vardi

Fumbling the Future

Fumbling the Future: How Xerox Invented, Then Ignored, the First Personal Computer is the title of a classic 1998 book by D.K. Smith and R.C. Alexander that tells the gripping story of how Xerox invented the personal-computing technology

in the 1970s, and then "miscalculated and mishandled" the opportunity to fully exploit it. To "fumble the future" has since become a standard phrase in discussions of advanced technology and its commercialization.

Another example definitely worth watching is a recently discovered copy of a 1993 AT&T commercial (http://www.geekosystem.com/1993-att-video/), with a rather clear vision of the future, predicting what was then revolutionary technology, such as paying tolls without stopping and reading books on computers. As we know today, the future did not work out too well for AT&T; following the telecom crash of the early 2000s, the telecom giant had to sell itself in 2005 to its former spin-off, SBC Communications, which then took the name AT&T.

This editorial is a story of how I fumbled the future. It is not widely known, but I *almost* invented the World-Wide Web—twice (smiley face here).

In 1988 I was a research staff member at the IBM Almaden Research Center. In response to a concern that IBM was not innovative enough, a group of about 20 researchers, including me, from different IBM Research labs, was tasked with envisioning an exciting information-technology-enabled 21st-century future. The "CS Future Work Group" met several times over a period of 18 months and produced a "vision" titled "Global Multi-Media Information Utilities." What did this somewhat clunky name refer to? To quote: "The utility is assumed to have a geographical coverage similar to that of today's telephone system, over which it can deliver, on a flexible pay-as-you-go basis, the most varied multimedia information services, such as multimedia electronic messaging, broadcasting, multimedia reference and encyclopedia database querying, rich information services, teleconferences, online simulation, visualization, and exploration services." In other words, while we did not predict Google, Facebook, or Wikipedia, we did describe, rather presciently I think, the World-Wide Web.

What happened with that exciting vision? Very little, I am afraid. Our report was issued as an IBM Research Report. It was deemed significant enough to be classified as "IBM Confidential," which ensured that it was not widely read, and it had no visibility outside IBM. The publication of the report practically coincided with the start of IBM's significant business difficulties in the early 1990s, and the corporation's focus naturally shifted to its near-term future. In other words, we fumbled the future. (At my request, the report was recently declassified; see https://researcher.ibm.com/researcher/files/zurich-pj/vision1%20-%202010.pdf.)

The future looks clear only in hindsight. It is rather easy to practically stare at it and not see it.

In the early 1990s, the computing environment at IBM Research was shifting from mainframe-based to workstation-based. Users requested an interface that would integrate the two environments. I got involved, together with several other people, in the development of ARCWorld, a software tool with a graphical user interface that allowed a user to manage and manipulate files on multiple Internet-connected computer systems. Our focus was on file manipulation, rather than information display, but ARCWorld could have been described as a "file browser." It was clear to all involved that ARCWorld was an innovative approach to wide-area information management, but it was difficult to see how it could fit within IBM's product strategy at the time. Some feeble attempts at commercialization were not successful. We fumbled the future, again.

What is the moral of these reminiscences? My main lesson is that fumbling the future is very easy. I have done it myself! The future looks clear only in hindsight. It is rather easy to practically stare at it and not see it. It follows that those who did make the future happen deserve double and triple credit. They not only saw the future, but also trusted their vision to follow through, and translated vision to execution. We should all recognize the incredible contributions of those who did *not* fumble the future.

Moshe Y. Vardi, EDITOR-IN-CHIEF

DOI:10.1145/1897852.1897854

Free Speech for Algorithms?

In "Regulating the Information Gatekeepers" (Nov. 2010), Patrick Vogl and Michael Barrett said a counterargument against the regulation of search-engine bias is that "Search results are free speech and therefore cannot be regulated." While I have no quarrel as to whether this claim is true, I'm astounded that anyone could seriously make such a counterargument—or any judge accept it.

Search results are the output of an algorithm. I was unaware the field of artificial intelligence had advanced to the point that we must now consider granting algorithms the right of free speech. To illustrate such absurdity, suppose I was clever enough to have devised an algorithm that could crawl the Web and produce opinionated articles, rather than search results, as its output. Would anyone seriously suggest the resulting articles be granted all the constitutional protections afforded the works of a human author? Taking the analogy further, suppose, too, my algorithm produced something equivalent to shouting "Fire!" in a crowded theater. Or, further still, perhaps it eventually produced something genuinely treasonous.

If we accept the idea that the output of an algorithm can be protected under the right of free speech, then we ought also to accept the idea that it is subject to the same limitations we place on truly unfettered free speech in a civilized society. But who would we go after when these limitations are exceeded? I may have created the algorithm, but I'm not responsible for the input it found that actually produced the offensive output. Who's guilty? Me? The algorithm? (Put the algorithm on trial?) The machine that executed the algorithm? How about those responsible for the input that algorithmically produced the output?

Unless humans intervene to modify the output of algorithms producing search results, arguments involving search results and free speech are absurd. At least until artificial intelligence has advanced to where machines must indeed be granted the same rights we grant our fellow humans.

Roger Neate, Seattle, WA

Authors' Response:

Neate touches a nerve concerning the increasingly complex relationship between humans and material technologies in society. Accountability in these socio-material settings is challenging for judge and regulator alike. In the 2003 case of SearchKing vs. Google Technology, *a U.S. District Court noted the ambiguity of deciding whether PageRank is mechanical and objective or subjective, ruling that PageRank represents constitutionally protected opinions. Whether search results are indeed free speech remains controversial, meaning we can expect the debate to continue.*

Patrick Vogl and Michael Barrett, Cambridge, U.K.

Science Has 1,000 Legs

It's great to reflect on the foundations of science in *Communications*, as in Tony Hey's comment "Science Has *Four* Legs" (Dec. 2010) and Moshe Y. Vardi's Editor's Letter "Science Has Only Two Legs" (Sept. 2010), but also how the philosophy of science sheds light on questions involving the number of legs in a natural science.

Willard Van Orman Quine's 1951 paper "Two Dogmas of Empiricism" convincingly argued that the attempt to distinguish experiment from theory fails in modern science because every observation is so theory-laden; for example, as a result of a Large Hadron Collider experiment, scientists will not perceive, say, muons or other particles, but rather some visual input originating from the computer screen displaying experimental data. The interpretation of this perception depends on the validity of many non-empirical factors, including physics theories and methods.

With computation, even more factors are needed, including the correctness of hardware design and the validity of the software packages being used, as argued by Nick Barnes in his comment "Release the Code" (Dec. 2010) concerning Dennis McCafferty's news story "Should Code Be Released?" (Oct. 2010).

For such a set of scientific assumptions, Thomas S. Kuhn coined the term "paradigm" in his 1962 book *The Structure of Scientific Revolutions*. Imre Lakatos later evolved the concept into the notion of "research program" in his 1970 paper "Falsification and the Methodology of Scientific Research Programs."

In this light, neither the two-leg nor the four-leg hypothesis is convincing. Citing the leg metaphor at all, science is perhaps more accurately viewed as a millipede.

Wolf Siberski, Hannover, Germany

Certify Software Professionals and their Work

As a programmer for the past 40 years, I wholeheartedly support David L. Parnas's Viewpoint "Risks of Undisciplined Development" (Oct. 2010) concerning the lack of discipline in programming projects. We could be sitting on a time bomb and should take immediate action to prevent potential catastrophic consequences of the carelessness of software professionals. I agree with Parnas that undisciplined software development must be curbed.

I began with structured programming and moved on to objects and now to Web programming and find that software is a mess today. When I travel on a plane, I hope its embedded software does not execute some untested loop in some exotic function never previously recognized or documented. When I conduct an online banking transaction, I likewise hope nothing goes wrong.

See the Web site "Software Horror Stories" (http://www.cs.tau.

ac.il/~nachumd/horror.html) showing why the facts can no longer be ignored. Moreover, certification standards like CMMI do not work. I have been part of CMMI-certification drives and find that real software-development processes have no relation to what is ultimately certified. Software development in real life starts with ambiguous specifications. When a project is initiated and otherwise unrelated employees assembled into a team, the project manager creates a process template and fills it with virtual data for the quality-assurance review. But the actual development is an uncontrolled process, where programs are assembled from random collections of code available online, often taken verbatim from earlier projects.

Most software winds up with an unmanageable set of bugs, a scenario repeated in almost 80% of the projects I've seen. In them, software for dropped projects might be revived, fixed by a new generation of coders, and deployed in new computer systems and business applications ultimately delivered to everyday users.

Software developers must ensure their code puts no lives at risk and enforce a licensing program for all software developers. Proof of professional discipline and competency must be provided before they are allowed to write, modify, or patch any software to be used by the public.

As suggested by Parnas,[1,2] software should be viewed as a professional engineering discipline. Science is limited to creating and disseminating knowledge. When a task involves creating products for others, it becomes an engineering discipline and must be controlled, as it is in every other engineering profession. Therefore, software-coding standards should be included in penal codes and country laws, as in the ones that guide other engineering, as well as medical, professions. Moreover, software developers should be required to undergo periodic relicensing, perhaps every five or 10 years.

Basudeb Gupta, Kolkata, India

References

1. Parnas, D.L. Licensing software engineers in Canada. *Commun. ACM 45*, 11 (Nov. 2002), 96–98.
2. Parnas, D.L. Software engineering: An unconsummated marriage. *Commun. ACM 40*, 9 (Sept. 1997), 128.

Unicode Not So Unifying

Poul-Henning Kamp's attack in "Sir, Please Step Away from the ASR-33!" on ASCII as the basis of modern programming languages was somewhat misplaced. While, as Kamp said, most operating systems support Unicode, a glance at the keyboard shows that users are stuck with an ASCII subset (or regional equivalent).

My dubious honor learning and using APL* while at university in the 1970s required a special "golf ball" and stick-on key labels for the IBM Selectric terminals supporting it. A vexing challenge in using the language was finding one the many Greek or other special characters required to write even the simplest code.

Also, while Kamp mentioned Perl, he failed to mention that the regular expressions made popular by that language—employing many special characters as operators—are virtually unintelligible to all but the most diehard fans. The prospect of a programming language making extensive use of the Unicode character set is a frightening proposition.

William Hudson, Abingdon, U.K.

*APL stands for "A Programming Language," so "the APL programming language" deconstructs as "the a programming language programming language."

The Merchant Is Still Liable

In his Viewpoint "Why Isn't Cyberspace More Secure?" (Nov. 2010), Joel F. Brenner said that in the U.K. the customer, not the bank, usually pays in cases of credit-card fraud. I would like to know the statistical basis for this claim, since for transactions conducted in cyberspace the situation in both the U.K. and the U.S. is that liability generally rests with the merchant, unless it provides proof of delivery or has used the 3-D Secure protocol to enable the card issuer to authenticate the customer directly. While the rates of uptake of the 3-D Secure authentication scheme may differ, I have difficulty believing that difference translates into a significant related difference in levels of consumer liability.

The process in the physical retail sector is quite different in the U.K. as a result of the EMV, or Europay, MasterCard, and VISA protocol, or "Chip & PIN," though flaws in EMV and hardware mean, in practice, the onus is still on the bank to demonstrate its customer is at fault.

Alastair Houghton, Fareham, England

Author's Response:

The U.K. Financial Services Authority took over regulation of this area November 1, 2009, because many found the situation, as I described it, objectionable. In practice, however, it is unclear whether the FSA's jurisdiction has made much difference. While the burden of proof is now on the bank, one source (see Dark Reading, *Apr. 26, 2010) reported that 37% of credit-card fraud victims get no refund. The practice in the U.S. is not necessarily better but is different.*

Joel F. Brenner, Washington, D.C.

Format Migration or Unforgiving Obsolescence

David S.H. Rosenthal's response (Jan. 2011) to Robin Williams' comment "Interpreting Data 100 Years On" said he was unaware of a single format widely used that has actually become obsolete. Though I understand the sentiment, it brought to mind Apple's switch from PowerPC to Intel architecture about six years ago. Upgrading the computers in my company in response to that switch required migrating all our current and legacy data to the new format used by Intel applications at the time. Though we didn't have to do it straightaway, as we could have kept running our older hardware and software, we had no choice but to commence a process to migrate over time.

This decision directly affected only my company, not the entire computing world, but when addressing data exchange and sharing, it was an additional factor we had to consider. Rather than face some general obsolescence, we may inevitably all have to address format obsolescence that is a natural consequence of IT's historically unforgiving evolution.

Bob Jansen, Erskineville, NSW, Australia

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

ACM, Advancing Computing as a Science and a Profession

Dear Colleague,

The power of computing technology continues to drive innovation to all corners of the globe, bringing with it opportunities for economic development and job growth. ACM is ideally positioned to help computing professionals worldwide stay competitive in this dynamic community.

ACM provides invaluable member benefits to help you advance your career and achieve success in your chosen specialty. Our international presence continues to expand and we have extended our online resources to serve needs that span all generations of computing practitioners, educators, researchers, and students.

ACM conferences, publications, educational efforts, recognition programs, digital resources, and diversity initiatives are defining the computing profession and empowering the computing professional.

This year we are launching Tech Packs, integrated learning packages on current technical topics created and reviewed by expert ACM members. The Tech Pack core is an annotated bibliography of resources from the renowned ACM Digital Library – articles from journals, magazines, conference proceedings, Special Interest Group newsletters, videos, etc. – and selections from our many online books and courses, as well an non-ACM resources where appropriate.

BY BECOMING AN ACM MEMBER YOU RECEIVE:

Timely access to relevant information

Communications of the ACM magazine • ACM Tech Packs • *TechNews* email digest • Technical Interest Alerts and ACM Bulletins • ACM journals and magazines at member rates • full access to the *acmqueue* website for practitioners • ACM SIG conference discounts • the optional ACM Digital Library

Resources that will enhance your career and follow you to new positions

Career & Job Center • The Learning Center • online books from Safari® featuring O'Reilly and Books24x7® • online courses in multiple languages • virtual labs • e-mentoring services • *CareerNews* email digest • access to ACM's 36 Special Interest Groups • an acm.org email forwarding address with spam filtering

ACM's worldwide network of more than 100,000 members ranges from students to seasoned professionals and includes many renowned leaders in the field. ACM members get access to this network and the advantages that come from their expertise to keep you at the forefront of the technology world.

Please take a moment to consider the value of an ACM membership for your career and your future in the dynamic computing profession.

Sincerely,

Alain Chesnais

President
Association for Computing Machinery

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: AD10

You can join ACM in several easy ways:

Online
http://www.acm.org/join

Phone
+1-800-342-6626 (US & Canada)
+1-212-626-0500 (Global)

Fax
+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/serving/ethics.html

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 ACM 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

10 Years of Celebrating Diversity in Computing

2011 Richard Tapia
Celebration of Diversity in Computing Conference

April 3-5, 2011 http://tapiaconference.org/2011/
San Francisco, CA

Register now to attend the 2011 Richard Tapia Celebration of Diversity in Computing Conference and save $75 – advance registration rates are valid through Tuesday, March 8.

Since 2001, the Tapia Celebration of Diversity in Computing has served as a leading forum for bringing together students, professors and professionals to discuss and strengthen their passion and commitment to computing. The 2011 program will include stellar speakers who are exemplary leaders in academia and industry, such as:

- **Irving Wladawsky-Berger,** former chair of the IBM Academy of Engineering and the 2001 HENAAC Hispanic Engineer of the Year, will give the Ken Kennedy Memorial Lecture on *"The Changing Nature of Research and Innovation in the 21st Century."*

- **Deborah Estrin,** the Jon Postel Professor of Computer Science at UCLA and a member of the National Academy of Engineering, will talk on *"Participatory Sensing: from Ecosystems to Human Systems."*

- **Alan Eustace,** Senior Vice President of Engineering and Research at Google, will give an after dinner talk entitled *"Organizing the World's Information."*

- **Ayanna Howard,** Associate Professor in the ECE School at Georgia Tech who *Technology Review* selected as a 2003 Young Innovator, will give the talk *"SnoMotes - Robotic Scientific Explorers for Understanding Climate Change."*

- **Blaise Aguera y Arcas,** Architect with Microsoft who was selected by *Technology Review* as a 2008 Young Innovator and was a celebrated speaker at the TED (Technology Entertainment Design) conference.

- **Illya Hicks,** Associate Professor in the Computational and Applied Mathematics Department at Rice University and recipient of the 2005 Optimization Prize for Young Researchers by the Optimization Society.

- **John Kubiatowicz,** Professor of Computer Science at the University of California, Berkeley, was selected by *Scientific American* in 2002 as one of 50 scientists for outstanding achievements in science and technology.

- **Patty Lopez,** Component Design Engineer with Intel, a winner of Hewlett Packard's Technical Leadership Award in 2001, and co-founder of Latinas in Computing.

Based on surveys from past attendees, for Tapia Conference 2011 we've added new programs to connect students with computing professionals, thereby opening the door to future opportunities, and a special outing to take in the sights of San Francisco. We will continue past popular sessions, including the Student Poster Session, Town Hall Meeting, Banquet, and the Doctoral Consortium, a day-long program designed to help equip students for the grueling challenge of finishing their doctorate. There will also be Resume, Grad School and Early Career Advice Workshops and attendee-proposed BOFs and panels.

The Conference program, news and registration information can be found at:
http://tapiaconference.org/2011/

Tapia Conference 2011 supporters include:
Google *and* ***National Science Foundation*** *(Platinum);*
Intel *(Gold);*
Cisco, Microsoft *and* ***NetApp*** *(Silver);*
IBM *and* ***Symantec*** *(Bronze);*
Amazon, Freddie Mac, Lawrence Berkeley National Laboratory, Lawrence Livermore National Laboratory, National Center for Atmospheric Research, National Security Agency *and* ***SAP*** *(Supporters).*

The Tapia Conference 2011 is organized by the Coalition to Diversify Computing and is co-sponsored by the Association for Computing Machinery and the IEEE Computer Society, in cooperation with the Computing Research Association.

DOI:10.1145/1897852.1897855

In the Virtual Extension

To ensure the timely publication of articles, Communications *created the Virtual Extension (VE) to expand the page limitations of the print edition by bringing readers the same high-quality articles in an online-only format. VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on merit. The following synopses are from articles now available in their entirety to ACM members via the Digital Library.*

viewpoint

DOI: 10.1145/1897852.1897880

Reaching Out to the Media: Become a Computer Science Ambassador

Frances Rosamond et al.

Science communication or public outreach can be seen as taking a lot of time and effort compared to the perceived payoffs these types of initiatives provide. In effect, there's a tragedy of the commons—we all benefit from those who do it, so there is incentive to let other people shoulder the load.

The rationale behind science communication is fairly obvious, and it is often difficult to provide compelling arguments that appeal to skeptics. Public outreach is related to the reputation of the scientific field, funding, and the integration of the science community into society. More locally and perhaps more relevantly, it is related to the reputation of your university and to the quality of your students.

Other sciences have established long-lasting traditions of transmitting their key issues, raising public awareness including highlights such as the Nobel Prize or the Fields Medal (however, rarely is the general public aware of the ACM A.M. Turing Award).

Computer science is not yet where it should be regarding public awareness. The reasons for this situation may lie in the relative youth of the area, the rapid advances in the field, as well as the fast-moving technology that computer science is related to. Computer scientists face the myriad drawbacks of lacking public awareness. They are confronted with low enrollment numbers and low funding, and to some extent, they feel ignored and misunderstood. The authors of this article provide suggestions for what can be pragmatically done to increase coverage of computer science in the media.

contributed article

DOI: 10.1145/1897852.1897881

The Internet Electorate

R. Kelly Garrett and James N. Danziger

The Internet was a prominent feature of the 2008 U.S. presidential election, regularly noted for its role in the Obama campaign's successful fundraising and supporter-mobilization efforts and for its widespread use by interested voters. This article reports on a national telephone survey conducted in the weeks following that election to assess how Americans' experience of elections was changing in response to the increasing availability and use of the digital communication network.

The Internet has long been heralded as an efficient means of acquiring political information, but the increasing presence of user-created content means the network is also becoming an important mode of political expression. The article examines these complementary roles, focusing on how Americans used the Internet to learn about the 2008 campaign, share political information, and voice their own opinions. Also considered is which individuals are most likely to engage in online information acquisition and expression, examining the influence of these practices on voters. Analyses are based on the national random-digit dial telephone survey of 600 adult Americans conducted two weeks after the 2008 election (November 6–20), with a response rate of 26.2%.

In the lead-up to the U.S. election in 2008, nearly two-thirds (64%) of Americans got campaign news online, a marked increase from 2004, when only about one-quarter (27%) of Americans said they got campaign news online. Equally notable is the fact that in 2008 two-fifths (38%) of respondents reported seeking online campaign news almost every day.

contributed article

DOI: 10.1145/1897852.1897882

Governing Web 2.0

Steven DeHertogh, Stijn Viaene, and Guido Dedene

Web 2.0 applications aspire to make maximal use of the level playing field for engagement offered by the Internet, both technologically and socially. The World Wide Web has thereby entered "the realm of sociality," where software becomes fused with everyday social life. This evolution has taken huge strides—Web 2.0 environments such as Wikipedia, Facebook, and MySpace have all become household names.

Both practitioners and researchers are converging on the usefulness of Web 2.0 for professional organizations. In and around enterprises, Web 2.0 platforms have been professed to support a profound change in intra- and inter-enterprise communication patterns. It is still early in terms of available management research on so-called "enterprise 2.0" experiences. Nevertheless, we have observed, as have others, that the way for organizations to capture benefits from Web 2.0 technology in the enterprise probably differs substantially from the way they attended to other enterprise information system projects in the past.

This article proposes a set of grounding principles to get the most out of enterprise 2.0 investments. The principles represent a synthesis of existing management theory and the author's own case research of companies with recent experience in introducing Web 2.0 into their enterprises. The successful introduction of Web 2.0 for the enterprise will require a move away from predesigned paternalistically imposed communication strategies and structures, toward carefully stimulating a many-to-many, decentralized emergence of bottom-up communicative connections.

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:10.1145/1897852.1897856 http://cacm.acm.org/blogs/blog-cacm

Scientists, Engineers, and Computer Science; Industry and Research Groups

Mark Guzdial discusses what scientists and engineers should know about computer science, such as Alan Kay's "Triple Whammy." Greg Linden writes about industry's different approaches to research and how to organize researchers in a company.

Mark Guzdial
"What do Scientists and Engineers Need to Know About Computer Science?"
http://cacm.acm.org/blogs/blog-cacm/96699

A new effort at the Texas Advanced Computing Center is aimed at teaching scientists and engineers about supercomputing. They argue that "Anyone looking to do relevant computational work today in the sciences and engineering must have these skills." They offer a certificate or portfolio in "Scientific Computation."

Greg Wilson has been going after this same goal using a different strategy. He suggests that before we can teach scientists and engineers about high-performance *computing*, we first have to teach them about computing. He leads an effort called "Software Carpentry" to figure out what to teach scientists and engineers about computing:

I've been trying to construct a better answer for the past 13 years; Software Carpentry (http://software-carpentry.org/blog/) is what I've arrived at. It's the 10% of software engineering (with a very small "e") that scientists and engineers need to know before they tackle GPUs, clusters, and other fashionable Everests. Like sanitation and vaccination, the basic skills it teaches are cheap and effective; unfortunately, the other characteristic they share is that they're not really what photo ops are made of. We've also found a lot of resistance based on survivor bias: all too often, senior scientists who have managed to get something to work on a supercomputer say, "Well, I didn't need version control or unit testing or any of that guff, so why should my students waste their time on it?" Most scientists rightly regard computing as a tax they have to pay in order to get results.

The evidence is that the problem of teaching *everyone else* about computer science is bigger than teaching *computer science majors* about computer science. Chris Scaffidi, Mary Shaw, and Brad Myers have estimated that, by 2012, there will be about three million professional software developers in the U.S., but there will also be about 13 million end-user programmers—people who program as part of their work, but do not primarily develop software. This result suggests that for every student in your computer science classes, there are four more students who could use some help in learning computer science. Those scientists and engineers who will be programming one day are in those other four.

Brian Dorn and I have a paper, "Learning on the Job: Characterizing the Programming Knowledge and Learning Strategies of Web Designers," in the 2010 ACM International Computing Education Research workshop on Brian's work studying graphic designers who program. Brian finds that these end-user programmers don't know a lot about computer science, and that lack of knowledge hurts them. He finds that they mostly learn to program through Google. In his most recent work, he is finding that not knowing much about computer science means that they're inefficient at searching. When they see "try-catch" in a piece of code that they're trying to understand, they don't know to look up "exception handling," and they can easily spend hours reading about Java

exception handling when they are actually working in JavaScript.

Maybe we should be teaching scientists and engineers about computer science more generally. But as Greg Wilson points out, they don't want much—they see computer science as a "tax." What's the *core* of computer science that even scientists and engineers ought to know? Alan Kay recently suggested a "Triple Whammy" (http://computinged.wordpress.com/2010/05/24/the-core-of-computer-science-alan-kays-triple-whammy/) defining the core of computer science:

1. Matter can be made to remember, discriminate, decide, and do.
2. Matter can remember descriptions and interpret and act on them.
3. Matter can hold and interpret and act on descriptions that describe anything that matter can do.

That's a pretty powerful set. It goes way beyond Python vs. Java, or using Perl to check genome sequences with regular expressions vs. using MATLAB for analyzing data from ecological simulations. How do we frame the Triple Whammy in a way that fledgling scientists and engineers would find valuable and learnable?

Reader's comment

The worrying trend I see is that many computer engineering graduates are interested in learning only a large set of programming languages, but dislike courses like algorithm design, not realizing that these languages are merely tools for implementing solutions. The end result is what you could call technicians but not engineers.

—Farhan Ahmad

Greg Linden
"Research in the Wild: Making Research Work in Industry"
http://cacm.acm.org/blogs/blog-cacm/97467

How to do research in academia is well established. You get grants to fund your group, attract students, publish papers, and pump out Ph.Ds. Depending on who you ask and how cynical they have become, the goals are some combination of impacting the field, educating students, and personal aggrandizement.

Research in industry is less established. How to organize is not clear. The purpose is not even well understood. The business strategy behind forming a research group sometimes seems to be little more than a variant of the Underpants Gnomes' plan in *South Park*. Phase 1: Hire Ph.Ds. Phase 2: ? Phase 3: Profit!

Generally, researchers in industry are supposed to yield some combination of long-term innovation, improving the sophistication of technology and products beyond simple and obvious solutions, and helping to attract talented and enthusiastic developers.

To take one example in search, without researchers who know the latest work, it would be hard for a company to build the thousands of classifiers that ferret out subtleties of query intent, document meaning, and spamminess, all of which is needed for a high-quality search experience. Information retrieval is a field that benefits from a long history of past work, and researchers often are the ones that know the history and how to stand on giants' shoulders.

Even so, there are many in industry that consider researchers an expensive luxury that their company can ill afford. Part of this comes from the historically common organizational structure of having a separate and independent research lab, which sometimes looks to be a gilded ivory tower to those who feel they are locked outside.

The separate research lab is the traditional structure, but a problematic one, not only for the perception of the group by the rest of the company, but also because the researchers can be so far removed from the company's products as to have little ability to make an impact. Many companies appear to be trying other ways of organizing researchers into the company.

For example, Google is well known for integrating many of its researchers into product groups and shifting them among product groups, working side-by-side with different development teams. While on a particular project, a researcher might focus on the part of the problem that requires esoteric knowledge of particular algorithms, but they are exposed to and work on many problems in the product. When this group comes together, everyone shares knowledge, and then people move to another group, sharing their knowledge again. Moreover, these ephemeral teams get people to know people, yielding valuable peer networks. When a tough research problem later comes up and no one nearby knows how to solve it, finding the person in the company who can solve it becomes much easier.

Many other companies, including Microsoft, Facebook, and Twitter, maintain separate research organizations, but try to keep the researchers working very closely with the product teams. At these companies, the impetus for novel research often is a problem in the product, usually a problem that would not be obvious in academia because of their lack of access to big data and scale.

What organizational structure works best in industry may depend on your goals. For immediate impact, having researchers integrated into product groups provides a lot of value; they are directly solving today's hard problems. But what about the problem that might hit in a year or two? And what about long-term breakthroughs, entirely new products, enabled by new technology no one has thought of yet?

My personal opinion leans mostly toward integrating researchers on projects, much like Google does, but also giving researchers 20% time (as all developers should get) and occasionally turning a 20% time project into a full project (again, as all developers should get, but the threshold for what is considered impactful might differ for a researcher, given the speculative gamble that is the nature of research). This strikes a balance between immediate impact, doing novel research, and taking advantage of a long-term opportunity when inspiration hits.

What do you think? How should researchers be organized in companies? Why?

Reader's comment

Research as a process and a profession and as a mind set is quite different than product-making. Pushing the two too close together or expecting people to be good at both may not always be optimal. See my "Research as product" post on the FXPAL Blog.

—Gene Golovchinsky

Mark Guzdial is a professor at the Georgia Institute of Technology. **Greg Linden** is the founder of Geeky Ventures.

DOI:10.1145/1897852.1897857 David Roman

Time to Change

The list of add-on features for *Communications*' Web site began to take form shortly after the site was launched two years ago, and was a starting point for revisions now under way. Over the course of the last several months, suggestions to enhance the site were solicited and explored and are now spiriting the changes that will lead to a streamlined and improved site later this year.

Always a work in progress, *Communications*' site has changed in some ways since launch, most visibly with the addition of author-generated videos on the homepage. Some backend changes have also taken place. The upcoming changes are intended to remove elements that some users find crowded or confusing.

Communications' 2010 Readership Survey, conducted by Harvey Research Inc., points out some of the site's favored features. Not surprisingly, the "Current Issue" is the most widely used feature, cited by 57.8% of survey respondents. The magazine archive, cited by 40.3%, ranks second. The News section ranks third, at 32.3%.

The readership survey also shows there is work to be done, particularly to encourage more frequent and longer visits to the site.

Participants in the survey were asked for suggestions to improve the site. It is impossible to implement all suggestions, though not for lack of time, interest, or resources, but for the wide variety of opinions expressed. For example, one 13-year veteran of ACM reads articles online only and asked ACM to "dispense with the print edition," while a 17-year member said the opposite. "I sit at the computer all day for work. I don't want to sit at it to read magazines." To accommodate such diverse preferences, both print and online formats will continue for the foreseeable future (http://cacm.acm.org/magazines/2011/2/104384).

Guiding these site plans are *Communications*' savvy Web Board, ACM volunteers, and HQ staffers, plus site testers and developers. All have a voice in the revised site. Plans are not finalized, but some upcoming changes will affect search functionality, social media features, and the addition of other ACM content. We look forward to announcing the next iteration of *Communications*' Web site in coming months.

ACM Member News

DAVID HAREL ELECTED TO ISRAEL ACADEMY OF SCIENCES AND HUMANITIES

David Harel, the William Sussman Professor of Computer Science and Applied Mathematics at the Weizmann Institute of Science, was inducted last December into the Israel Academy of Sciences and Humanities. He is the first Academy member to be inducted as a computer scientist. Several computer scientists, including Noga Alon, Amir Pnueli, Michael Rabin, and Adi Shamir, had been previously inducted into the Academy, but as mathematicians. In the past Harel has worked in several areas of theoretical computer science, but in recent years his research has focused on areas such as software and systems engineering, visual languages, and the modeling and analysis of biological systems, and taken a more practical direction. At the induction ceremony, Harel says, the Academy's head of natural sciences made a point of telling Harel that he was "happy to have me in because of my practical work."

In January, Harel was honored by the A.M.N. Foundation for the Advancement of Science, Art, and Culture in Israel, which named him one of seven recipients of the $1 million EMET Prize. Harel was recognized for his studies on a "wide variety of topics within computer science, for his results that are at the forefront of scientific research, and for his achievements that have become a standard and working tools in many industries around the world."

Harel suggests his selection by the Academy may bring greater recognition to computer science and "make it easier to get people in on this ticket." But "far more important to me," Harel says, is that Israel's three A.M. Turing Award winners—Pnueli, Rabin, and Shamir—were previously elected into the Academy. "This makes me extremely proud and humbled by my own election. Great company indeed!"

—David Lindley

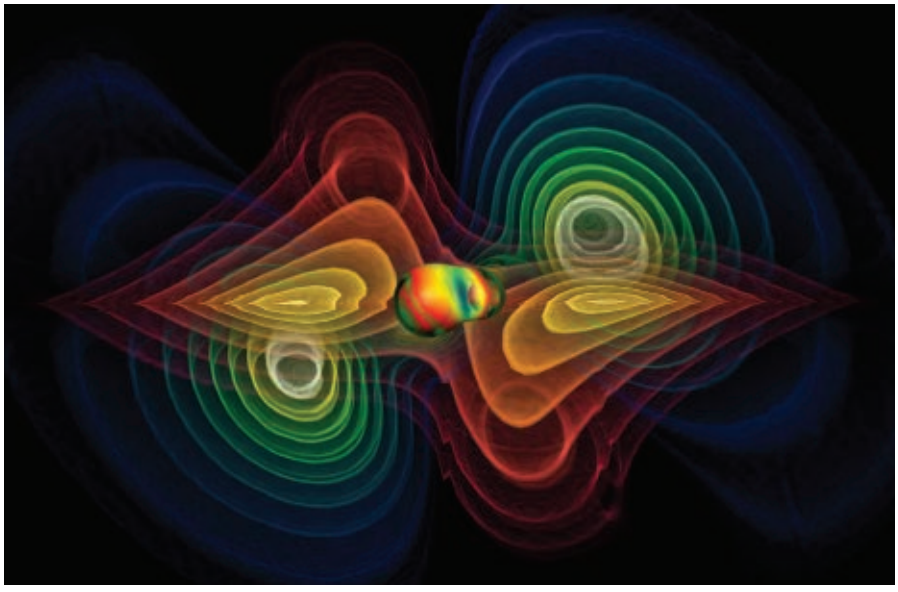
Science | DOI:10.1145/1897852.1897858 Kirk L. Kroeker

Grid Computing's Future

Outreach programs and usability improvements are drawing many researchers to grid computing from disciplines that have not traditionally used such resources.

In recent years, several powerful research grids consisting of thousands of computing nodes, dozens of data centers, and massive amounts of bandwidth have emerged, but few of these grids have received much attention in the mainstream media. Unlike seti@home, folding@home, and other highly focused grid projects that have captured the popular imagination by allowing home users to donate compute cycles, the big research grids are not accessible to the public and their fame does not extend far beyond the researchers who use them. Outreach teams and usability engineers at the largest of these new grids, such as Naregi, Egee, and TeraGrid, are trying to change that reality by helping to facilitate the adoption of grid technologies in fields that have not traditionally used grid-based supercomputing resources.

TeraGrid, said to be the world's largest distributed network infrastructure for open scientific research, is one such network that has quietly been making waves in research communities outside computer science, and is helping to solve complex problems in biology, physics, medicine, and numerous other fields. TeraGrid consists of 11 datacenter sites located around the U.S. and is tied to a 10-gigabyte backbone that connects primary network facilities in Los Angeles, Denver, and Chicago. At maximum capacity, TeraGrid can produce more than a petaflop of computing power and store more than 30 petabytes of data. The project, started by the National Science Foundation in August 2001, has grown in the past five years from fewer than 1,000 active users in 2005 to nearly 5,000 active users working on almost 1,600 projects at the end of 2009.



A grid-based computer simulation of the gravitational waves produced as two black holes merge with each other to form a larger black hole.

SCIENTIFIC VISUALIZATION BY WERNER BENGER, AEI/ZIB/LSU/UIBK

Matthew Heinzel, director of TeraGrid's grid infrastructure group, says that TeraGrid's outreach teams have done an excellent job drawing researchers from fields outside computer science. To obtain CPU time on the grid, scientists simply submit a request that describes the work to be done; extensive CPU-time requests are subject

Computers in Scientific Research

In 2009, Greg Wilson published details of a research project designed not only to determine how scientists learn what they know about using computers and how widely they share their software, but also to gauge the ratios of scientists who use desktop systems versus local clusters and grid-computing resources. Conventional wisdom suggests that as clusters and grids become more sophisticated and increasingly user friendly, more scientists will use them instead of less powerful desktop PCs. The findings Wilson and his colleagues produced run counter to that thinking.

While hard numbers indicate that more scientists each year are using grid-based resources to conduct their research, Wilson's project indicates that amount of work is relatively small compared to the vast number of projects still being run on the desktop. Some 81% of those participating in the research said they primarily use desktop machines, with only 13% saying they use intermediate-sized machines such as local clusters, and 6% saying they use supercomputers for their research.

Wilson, until recently a computer science professor at the University of Toronto and now at work full time on a course designed to address what he perceives as a computer-skills shortfall among scientists, says these figures might very well change in the years ahead. "If we're lucky, though, most scientists won't notice," he says. "Behind the scenes, an increasing number of data services will migrate upward, but the vast majority of scientists will reach them through desktop interfaces, because that's all they'll have the skills for."

That trend might already be materializing. Wilson says that in the year since the details of his research were published, he has witnessed an increased interest in commodity clouds, such as Amazon's Elastic Compute Cloud. "The course Titus Brown just taught at Michigan State [http://bit.ly/aE5Qpg] would have been a lot harder on everyone without pay-and-play," says Wilson, who predicts that more researchers will begin using such systems, which can be run independently from supercomputing-style applications.

As for the efforts to make grids easier to use and to draw researchers from fields that don't typically train scientists in supercomputer programming, Wilson says these efforts will fail outright unless scientists first invest in basic computing skills. "Asking someone who doesn't know how to modularize or test a program to use clouds and GPUs is as sensible as taking someone who's just learning how to drive a family car and putting them behind the wheel of a transport truck on the highway at rush hour," he says.

The best solution, according to Wilson, is to teach scientists some basic computational skills so they can tackle the challenges posed by advanced resources without expending heroic effort, which is the primary goal of the project Wilson is working on now, called Software Carpentry. As for future research to explore ways of assessing the impact of computers on scientists' productivity, Wilson wasn't able to find funding. "This is one of the reasons I left academia and returned to industry," he says. "As in health care, it seems that most people are more interested in pushing new things to market than in finding out what works and what doesn't." —*Kirk L. Kroeker*

to a quarterly peer-review process. Surprisingly, molecular biosciences, physics, astronomical sciences, chemistry, and materials research top the list of disciplines whose researchers are using the grid. "In some cases, we are victims of our own success," says Heinzel. "We've extended our user base well beyond our goals." But the downside to this success, he says, is that TeraGrid is being asked for far more CPU cycles than it can provide.

Still, Heinzel says that having more demand than capacity is a positive force that generates an impetus for the network to grow regularly. But the challenges facing Heinzel and TeraGrid's infrastructure group, which is run through the University of Chicago in partnership with the Argonne National Laboratory, extend beyond mere compute cycles and bandwidth. As more users from outside computer science are drawn to TeraGrid to run complex computational problems, one ongoing and key challenge is usability for those who do not have the necessary computer science skills. (The sidebar "Computers in Scientific Research," above, describes one project designed to address this skills shortfall.)

TeraGrid's resources, coordinated by UNIX-based Globus grid software, are integrated through a service-oriented architecture, with all systems running a common user interface. While the interface includes job-submission and data-management tools, the network requires problems to be translated into specific types of grid-ready code. To accommodate those who do not have the skills needed to translate their problems into compatible code, TeraGrid has allocated funding to embed dedicated supercomputer specialists in research teams for up to one year. Heinzel says that these specialists, who write and refine project-specific code, are among TeraGrid's most valuable resources, and represent a large part of the infrastructure group's budget.

"You can't just put a big piece of hardware on the floor and say, 'OK guys, here you go,' " says Heinzel. "You need somebody with the skills to help people not only run the code but also improve the code."

Modeling Black Holes

One researcher who has conducted extensive work on TeraGrid is Erik Schnetter, an assistant research professor in the department of physics and astronomy at Louisiana State University. His research has modeled black holes, neutron stars, and other highly complex astrophysical objects. The most interesting of these objects, he says, are gamma-ray bursts, which are bright bursts of high-energy photons that are visible from Earth and are said to be generated by the most energetic events in the universe. "It turns out that these bursts emanate from billions of light years away, essentially at the other end of the universe," says Schnetter. "The fact that they are still so brightly visible here means that they must come from truly tremendous explosions."

The mechanism that creates these explosions, the source of the energy, is not completely understood. After decades of research, the astrophysics community found that one model, called the collapsar model, might help explain the gamma-ray bursts. "What we do in one of our projects is model stars that form a supernova, then form a black hole at their center, and then we study how the remaining material behaves," says Schnetter. "These are very complex systems, and modeling them is a large task."

The computer code used to calculate these models not only is complex, but also requires significant computational power. Schnetter's group has performed simulations on local workstations and clusters, but he says that any kind of production work that has a

high level of accuracy requires systems that are too large for a single university. For example, the nodes processing Schnetter's modeling code require communication with other nodes several times per second and a data-exchange rate of about one gigabyte per second.

"To finish a simulation in a reasonable time, we need to use hundreds or thousands of cores," he says. "That means we need to split the problem into many pieces, and we need to ensure that each of these pieces remains as independent from the others as possible." By using this modular technique, Schnetter's team can replace or exchange grid code if it becomes necessary to apply new physics or use a different hardware architecture.

Another project run on TeraGrid is an effort to understand the environmental impact of aviation. Conducted out of Stanford University by doctoral candidate Alexander Naiman and overseen by Sanjiva Lele, a professor in the department of aeronautics and astronautics, the project models condensation trails, the ice clouds formed by aircraft emissions. Naiman, whose research group specializes in computational fluid dynamics and turbulence simulations, says the difficulty of contrail modeling becomes increasingly acute as the complexity of the model increases. "The more complex the flow, the higher resolution required to simulate it, and the more resources are needed," he says.

While Stanford has local supercomputing resources, they are in high demand. "TeraGrid provides relatively large and modern supercomputing resources to projects like ours that have no other supercomputing support," he says. The simulation code that Naiman and his team run on TeraGrid was written at the Center for Turbulence Research at Stanford. Naiman says it was easy to get that code running on TeraGrid. The research group parallelized the program, a type of large eddy simulation, using standard message-passing interface strategies that Naiman says have been highly scalable on TeraGrid.

The contrail modeling project is ongoing, but so far the Stanford team has simulated the first 20 minutes of contrail development for several scenarios, producing terabytes of three-dimensional flow fields and other data, such as time histories for cloud ice mass. Naiman says that although the TeraGrid data is still undergoing analysis, it is likely to help improve understanding of the development of contrails and their environmental impact. "TeraGrid performs as advertised, providing us with CPU hours that we would not have had access to otherwise," he says. "We also take advantage of the large archival storage available on TeraGrid to ensure that important data is backed up."

Improving Grid Software

As for the future of research on grid networks, TeraGrid's Heinzel says he remains optimistic, but points out that improvements must be made in grid software not only to enhance ease of use for researchers such as Schnetter and Naiman, but also to take complete advantage of new generations of hardware. "You have to be almost a systems admin to set your parameters on data movement correctly so you can take full advantage of these systems," says Heinzel. "So the software really needs to mature."

Echoing these concerns, LSU's Schnetter points out that his research groups consist of people with widely varying degrees of supercomputer experience. "Teaching everybody how to use the different systems, and staying on top of what works best on what system, and which parameters need to be tuned in what way to achieve the best performance, is like herding cats," he says. "There are almost no GUIs for supercomputers, and most of the ones that exist are really bad, so that using them requires some arcane knowledge."

Schnetter says he hopes that grid-based supercomputing will have a much larger influence on the curriculum than it does today, especially with so few universities teaching scientific programming at the level required to effectively use grid resources. "The good students in my group learned programming by themselves, on the side, because they were interested," he says. Still, Schnetter suggests that such self-taught programming might not be sustainable in a world in which computers are becoming increasingly complex. "I hope that this changes in the next decade," he says.

Stanford's Naiman offers a similar observation. He says that while his work on the grid has been positive, the usability of the technology could be improved. For his part, TeraGrid's Heinzel says he remains optimistic about the accessibility of grids, and predicts that major usability improvements are on the way. He likens the evolutionary pace of these grid developments to how the Web quickly emerged from the Internet and now requires little more than a browser and a basic knowledge of hyperlinks. "If you know exactly how to use grid tools, they work effectively," says Heinzel. "Now we need to make them more user-friendly so we can get a wider audience."

In the future envisioned by Heinzel, grids will be manipulated easily by computer scientists while still providing friendly interfaces for researchers coming from other fields. Rather than predicting that the arrival of such technologies will take decades or more, Heinzel says that much progress will be made in the next few years alone. "We're going to see some big improvements in the usability of the grid and grid software in the next two to four years," he says. "Future systems will be very user-friendly with a high degree of abstracting the inner workings of what's going on from the end users." C

Further Reading

Ferreira, L., Lucchese, F., Yasuda, T., Lee, C.Y., Queiroz, C.A., Minetto, E., and Mungioli, A.S.R. ***Grid Computing in Research And Education.*** **IBM Redbooks, Armonk, NY, 2005.**

Magoulès, F.
Fundamentals of Grid Computing: Theory, Algorithms, and Technologies. **Chapman & Hall, Boca Raton, FL, 2009.**

Neeman, H., Severini, H., Wu, D., and Kantardjieff, K.
Teaching high performance computing via videoconferencing, *ACM Inroads 1*, 1, March 2010.

Scavo, T., and Welch, V.
A grid authorization model for science gateways, *International Workshop on Grid Computing Environments 2007*, Reno, NV, Nov. 11, 2007.

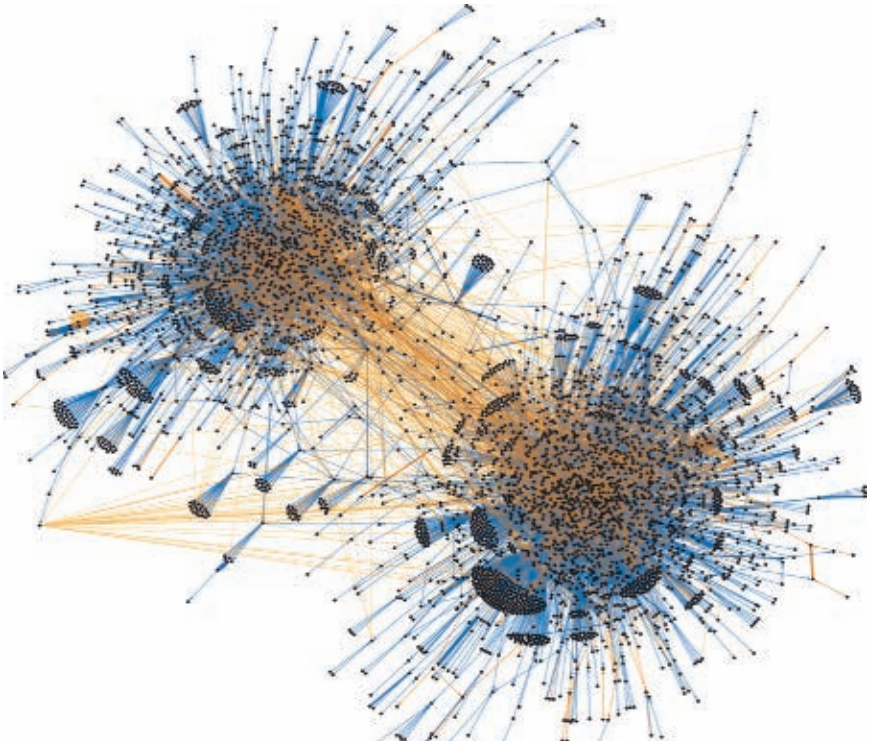
Wong, J. (Ed.)
Grid Computing Research Progress. **Nova Science Publishers, Hauppauge, NY, 2008.**

Based in Los Angeles, **Kirk L. Kroeker** is a freelance editor and writer specializing in science and technology.

Society | DOI:10.1145/1897852.1897860 Neil Savage

Twitter as Medium and Message

Researchers are mining Twitter's vast flow of data to measure public sentiment, follow political activity, and detect earthquakes and flu outbreaks.



Truthy shows how a tweet propagates, with retweets in blue and topic mentions in orange. Tweets that are sent back and forth between two Twitter accounts appear as a thick blue bar.

IMAGE COURTESY OF TRUTHY.INDIANA.EDU

Twitter generates a lot of noise. One hundred sixty million users send upward of 90 million messages per day, 140-character musings—studded with misspellings, slang, and abbreviations—on what they had for lunch, the current episode of "Glee," or a video of a monkey petting a porcupine that you just have to watch.

Individually, these tweets range from the inane to the arresting. But taken together, they open a surprising window onto the moods, thoughts, and activities of society at large. Researchers are finding they can measure public sentiment, follow political activity, even spot earthquakes and flu outbreaks, just by running the chatter through algorithms that search for particular words and pinpoint message origins.

"Social media give us an opportunity we didn't have until now to track what everybody is saying about everything," says Filippo Menczer, associate director of the Center for Complex Networks and Systems Research at Indiana University. "It's amazing."

The results can be surprisingly accurate. Aron Culotta, assistant professor of computer science at Southeastern Louisiana University, found that tracking a few flu-related keywords allowed him to predict future flu outbreaks. He used a simple keyword search to look at 500 million messages sent from September 2009 to May 2010. Just finding the word "flu" produced an 84% correlation with statistics collected by the U.S. Centers for Disease Control and Prevention (CDC). Adding a few other words, like "have" and "headache" increased the agreement to 95%.

The CDC's counts of what it terms influenza-like illness are based on doctors' reports of specific symptoms in their patients, so they're probably a more accurate measure of actual illness than somebody tweeting "home sick with flu." But it can take a week or two for the CDC to collect the data and disseminate the information, by which time the disease has almost certainly spread. Twitter reports, though less precise, are available in real time, and cost a lot less to collect. They could draw health officials' attention to an outbreak in its earlier stages. "We're certainly not recommending that the CDC stop tracking the flu the way they do it now," Culotta says. "It would be nice to use this as a first-pass alarm."

Twitter data may help answer sociological questions that are otherwise hard to approach, because polling enough people is too expensive and time consuming.

Google Flu Trends does something similar. One potential point in Twitter's favor is that a tweet contains more words, and therefore more clues to meaning, than the three or four words of a typical search engine query. And training algorithms to classify messages—filtering out the tweets that talk about flu shots or Bieber Fever—improves the accuracy further.

There are other physical phenomena where Twitter can be an add-on to existing monitoring methods. Air Twit-

ter, a project at Washington University in St. Louis, looks for comments and photos about events like fires and dust storms as a way to get early indications about air quality. And the U.S. Geological Survey (USGS) has explored using Twitter messages as a supplement to its network of seismographic monitors that alert the federal agency when an earthquake occurs. Paul Earle, a seismologist at the USGS, is responsible for quickly getting out information about seismic activity. He has searched for spikes in keywords—"OMG, earthquake!" is a popular tweet—for a quick alert of an event. "It's another piece of information in the seconds and minutes when things are just unfolding," Earle says. "It comes in earlier. Some or most of its value is replaced as we get more detailed or science-derived information."

Earle says Twitter might help weed out the occasional false alarm from automated equipment, when no tweets follow an alert. The content of tweets might also supplement Web-based forms that collect people's experiences of an earthquake and are used to map the event's intensity, a more subjective measure of impact that includes factors such as building damage. A recent earthquake in Indonesia, for instance, produced a spike of tweets—in Indonesian. There's no Web form in that language for intensity, but Earle says Indonesian tweets might help fill in the blanks.

Sentiment Analysis

Many researchers are doing sentiment analysis of tweets. Using tools from psychology, such as the Affective Norms for English Words, which rates the emotional value of many words, Alan Mislove tracked national moods, and found that Americans tend to be a lot happier on Sunday morning than Thursday evening, and that West Coast residents seem happier than those on the East Coast.

"I think this is going to be one of the most important datasets of this era, because we are looking at what people are talking about in real time at the scale of an entire society," says Mislove, an assistant professor of computer science at Northeastern University. He says there's no easy way to validate those results, but as a proof-of-concept it shows

"If you think about our society as being a big organism," says Noah Smith, "this is just another tool to look inside of it."

the sorts of information that might be derived from the Twitter data stream, even without taking additional steps to filter out false positives. "There's just simply so much data that you can do pretty decently, even by taking naïve approaches," says Mislove.

This type of inquiry, of course, has limitations. Researchers readily admit that Twitter data is noisy, and it's not always simple to know what a word means—in some parlances, "sick" is a good thing. But with hundreds of millions of messages, the errors tend to shrink. Another worry is hysteria; people worried about swine flu might tweet about it more, leading others to worry and tweet (or retweet), so there's a spike in mentions without any increase in actual cases.

There's also sample bias; certain segments of the population use Twitter more than others. But researchers seeking to glean insights from tweets can apply corrections to the sample, just as traditional pollsters do. And as a wider variety of people send more tweets, the bias is reduced. "The more data you have, the closer you get to a true representation of what the underlying population is," says Noah Smith, an assistant professor of computer science at Carnegie Mellon University.

Smith is examining how Twitter can supplement more familiar polling. One advantage is that pollsters can influence the answers they get by the way they phrase a question; people are fairly consistent, for example, in being more supportive of "gay marriage" than of "homosexual marriage," just because of the word choice. Studying tweets, which people send out of their own accord, removes that problem. "We're not actually talking to anyone. We're not

Technology

IBM's 2015 Predictions

IBM recently unveiled its fifth annual "Next Five in Five," a list of technology innovations the company says have the potential to change how people work, live, and play over the next five years. The IBM predictions are:

- **3D interfaces will let people interact via 3D holograms in real time. As 3D and holographic cameras become more sophisticated and miniaturized to fit into mobile phones, users will be able to interact with photos, surf the Web, and chat in novel ways.**
- **Scientific advances in transistors and battery technology will allow devices to last about 10 times longer than they do now. Instead of today's heavy lithium-ion batteries, scientists are working on batteries that use air to react with energy-dense metal.**
- **Sensors in phones, cars, and other objects will collect data to give scientists a real-time picture of the environment. IBM recently patented a technique that enables a system to accurately conduct post-event analysis of seismic events, as well as provide early warnings for tsunamis.**
- **Advanced analytics technologies will provide personalized recommendations that get commuters where they need to go in the fastest time. Using new mathematical models and IBM's predictive analytics technologies, researchers will analyze and combine multiple possible scenarios that can deliver the best routes for daily travel.**
- **Innovations in computers and data centers are enabling the excessive heat and energy that they give off to help heat buildings in the winter and power air conditioning in the summer. With new technologies, such as novel on-chip water-cooling systems, the thermal energy from processors can be efficiently recycled to provide hot water for an office or home.**

The 2015 predictions are based on emerging technologies from IBM's labs around the world as well as market and societal trends.

—Bob Violino

asking a question. We're just taking found data," Smith says. "We can get a much larger population of people participating passively."

Indeed, he says, Twitter data may help researchers answer all sorts of sociological questions that are otherwise hard to approach, because polling enough subjects is too expensive and time-consuming using traditional methods. For instance, Smith says, a researcher might study how linguistic patterns correlate to socioeconomic status, and perhaps learn something about communication patterns among people in different demographic groups. That, in turn, could reveal something about their access to information, jobs, or government services.

Of course, the power of widespread, unfiltered information invites the possibility of abuse. Two researchers at Wellesley University, Panagiotis Metaxas and Eni Mustafaraj, found that during a special election in Massachusetts for U.S. Senate, the Democratic candidate, Martha Coakley, was the subject of a "Twitter bomb" attack. A conservative group in Iowa, the American Future Fund, sent out 929 tweets in just over two hours with a link to a Web site that attacked Coakley. The researchers estimate the messages could have been seen by more than 60,000 people before being shut down as spam.

Indiana's Menczer developed a tool to distinguish between organized partisan spamming and grass-roots activism. He calls it Truthy, from comedian Stephen Colbert's coinage describing a statement that sounds factual but isn't. Starting with a list of keywords that includes all candidates, parties, and campaign names, the system detects what he calls memes, messages about a specific topic or candidate. It then displays a graphic representation of how each meme propagates, with retweets in blue and mentions of the topic in orange. If someone sets up two accounts and repeatedly sends the same tweets back and forth between them—an effort to show up in Twitter's popular "trending topics" list—it appears as a thick blue bar. Networks of automated tweets pushing a particular meme show up as regular, orange starbursts. More natural propagations look like fuzzy dandelions. In some cases, the tweets carry links to Web sites with questionable claims or even strident propaganda. Others turn out to be pitching a product.

The patterns, along with information about when each Twitter account was created and whether the owner is known, allow voters to distinguish actual political dialogue from organized attacks. Menczer hopes to add sentiment analysis to analyze the content of the messages as well as their dispersal patterns.

At Xerox Palo Alto Research Center, Research Manager Ed H. Chi is also looking at message propagation. "Twitter is kind of this perfect laboratory for understanding how information spreads," Chi says. Such a study can improve theoretical models of information dispersal, and also give people and businesses better strategies for delivering their messages or managing their reputations.

Much Twitter-based research is still preliminary. Some findings can be validated through other sources, such as CDC statistics or public opinion polls, but others remain unproven. Still, the scientists are excited at the prospects of what they might find by mining such a large, raw stream of data. "As Twitter and other social media grow, you'll be able to ask much more fine-grained questions," says Smith. "If you think about our society as being a big organism, this is just another tool to look inside of it."

Further Reading

Chen, J., Nairn, R., Nelson, L., and Chi, E. H.
Short and tweet: experiments on recommending content from information streams, ACM Conference on Human Factors in Computing Systems, Atlanta, GA, April 10–15, 2010.

Culotta, A.
Detecting influenza outbreaks by analyzing Twitter messages, KDD Workshop on Social Media Analytics, Washington, D.C., July 25, 2010.

Earle, P., Guy, M., Buckmaster, R., Ostrum, C., Horvath, S., and Vaughan, A.
OMG earthquake! Can Twitter improve earthquake response? *Seismological Research Letters 81*, 2, March/April 2010.

Metaxas, P.T. and Mustafaraj, E.
From obscurity to prominence in minutes: political speech and real-time search, Web Science Conference, Raleigh, NC, April 26–27, 2010.

O'Connor, B., Balasubramanyan, R., Routledge, B., and Smith, N.
From Tweets to polls: linking text sentiment to public opinion time series, *Proceedings of the International AAAI Conference on Weblogs and Social Media*, Washington, D.C., May 23–26, 2010.

Neil Savage is a science and technology writer based in Lowell, MA.

Milestones

AAAS Fellows

In December, the American Association for the Advancement of Science (AAAS) elected 503 members as Fellows in recognition of their meritorious efforts to advance science or its applications. Election as an AAAS Fellow is an honor bestowed upon members by their peers.

Of the new Fellows, 16 members were selected for the Section on Information, Computing, and Communication. They are: Srinivas Aluru, Iowa State University; Victor Bahl, Microsoft Research; David R. Boggs, Consulting Electrical Engineer; Geoffrey Charles Bowker, University of Pittsburgh; John M. Carroll, Pennsylvania State University; J. J. Garcia-Luna-Aceves, University of California, Santa Cruz/Palo Alto Research Center; Venu Govindaraju, University at Buffalo State, The University of New York; Hamid Jafarkhani, University of California, Irvine; Farnam Jahanian, University of Michigan; Phokion G. Kolaitis, University of California, Santa Cruz; C. C. Jay Kuo, University of Southern California; Dinesh Manocha, University of North Carolina, Chapel Hill; Hanan Samet, University of Maryland; Abraham Silberschatz, Yale University; Manuela M. Veloso, Carnegie Mellon University; and Barry Wessler, Wessler Consulting.

The new Fellows were honored at the Fellows Forum held in February during the AAAS Annual Meeting in Washington, D.C.

Research and Development | DOI:10.1145/1897852.1897861 Tom Geller

Evaluating Government Funding

Presidential report asserts the value of U.S. government funding and specifies areas needing greater focus.

President Obama enjoys a lighthearted moment with members of the President's Council of Advisors on Science and Technology during a meeting at the White House.

Is computer science rightly part of public education? How much does the U.S. government spend on basic networking and IT research? Should industry provide that funding instead? How important is supercomputing?

These are some of the questions addressed by a 148-page report released by the President's Council of Advisors on Science and Technology (PCAST) last December. Titled *Designing a Digital Future: Federally Funded Research and Development in Networking and Information Technology*, the report looked at U.S. investments in the cross-agency Networking and Information Technology Research and Development (NITRD) program, currently totaling approximately $4.3 billion per year.

Among other points, the council called for affirmation of computer science as a part of education in science, technology, engineering, and math; increased investment in the areas of privacy, human-computer interaction, massive data stores, and physical instrumentation such as sensors and robotics; long-term, multi-agency NIT initiatives for health, energy, transportation, and security; better coordination among agencies by the Office of Science and Technology Policy and the National Science and Technology Council; and a standing committee to provide ongoing strategic perspectives.

The report also warned against single-minded performance metrics when evaluating high-performance computing (HPC) projects—a subject made timely by Top500's most-recent ranking of the world's fastest supercomputers, which appeared five weeks before PCAST released its report. The Chinese-built Tianhe-1A supercomputer topped that list, bumping U.S.-made computers from the lead spot for the first time in six years. The report stated that "comparative rankings of the world's fastest supercomputers" are "relevant to only some of our national priorities," and said that they shouldn't " 'crowd out' the fundamental research in computer science and engineering that will be required to develop truly transformational next-generation HPC systems."

PCAST called for an increase of $1 billion in funding for "new, potentially transformative NIT research" and recommended more specific accounting to separate basic NIT research from infrastructure costs. The report's Working Group Co-chair and University of Washington Professor Ed Lazowska was quick to point out that the money was used in ways that are "appropriate and important"—for example, large-scale genome databases—although not "pushing the forefront of NIT."

Independent technology reviews of this sort were mandated under the High-Performance Computing Act of 1991. The previous review, published in 2007, "found many of the same issues" according to that report's co-chair, Microsoft Corporate Vice President Daniel Reed. But he did note some changes of focus, such as the marked increase in data. "We've gone from a world where data was rare and precious to where we're drowning in it," Reed says.

The report also documented NIT's importance to U.S. competitiveness—and the payback for NIT investment.

An example of high payback was given at the report's public release by Akamai founder Tom Leighton. He related a story of U.S. Defense Advanced Research Projects Agency funding he received in the 1990s to study "highly mathematical and highly theoretical subjects ... the living example of high-risk research." When the research was finished, Internet companies weren't interested in the results—even for free.

"So we started a company called Akamai Technologies," Leighton says. "We [now] carry over a third of Web traffic ... and are probably paying over a $100 million in taxes this year. But it wasn't the kind of research that companies fund."

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

OFFICIAL WHITE HOUSE PHOTO BY PETE SOUZA

Technology | DOI:10.1145/1897852.1897859 Gary Anthes

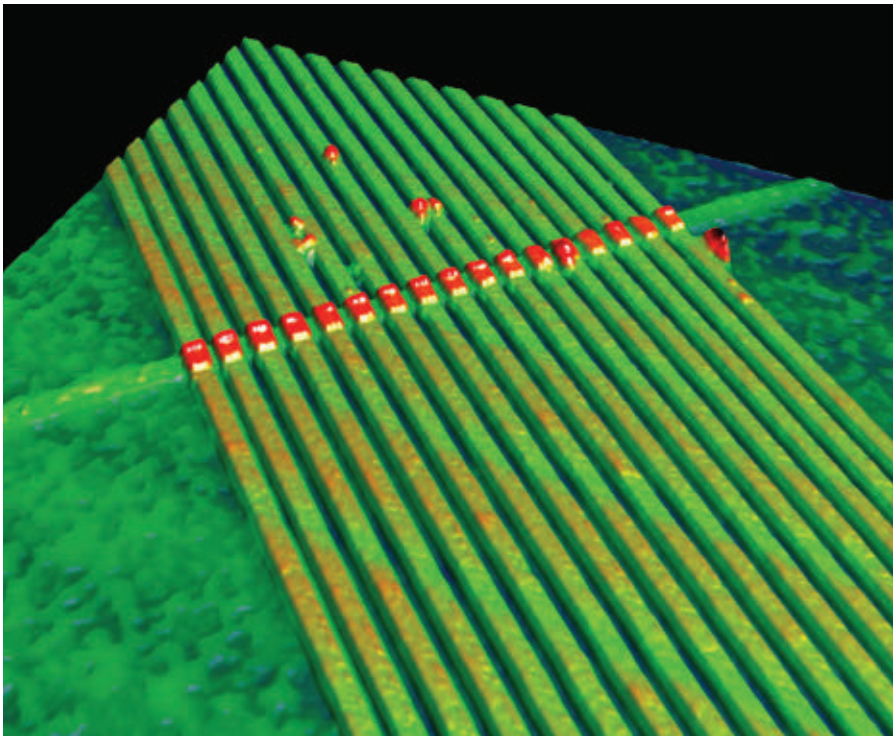
Memristors: Pass or Fail?

The device may revolutionize data storage, replacing flash memory and perhaps even disks. Whether they can be reliably and cheaply manufactured, though, is an open question.

A FUNDAMENTAL ELECTRONIC device, whose existence was postulated five decades ago but which proved hard to understand, let alone build, is ready to emerge from the lab, corporate and university researchers say. If so, the memristor (or memory resistor), as it is called, may arrive just in time to save the information storage industry from the transistor's collision with the scaling wall at the end of Moore's Law.

Hewlett-Packard announced last August that it would team with the South Korean computer memory maker Hynix Semiconductor to develop memristor-based memory chips, called resistive RAM (ReRAM), which they say will be on the market in about three years. The companies say their titanium-based chips could replace flash memory—which has become nearly ubiquitous in mobile applications—and would be 10 times faster and 10 times more energy efficient. Meanwhile, Rice University has joined with Austin, TX-based PrivaTran, a semiconductor design company specializing in custom integrated systems, to develop an all-silicon ReRAM chip that could be a substitute for flash memory. But a senior research official at Intel says it is far from certain that either effort will succeed.

A memristor is a tiny two-terminal electronic component that can be made from a variety of materials—including polymers, metal oxides, and conventional semiconductors like silicon— whose resistance varies with the voltage applied across it and with the length of time the voltage is applied. Its initial applications are likely to be as binary memory devices, but it could work in an analog fashion and could eventually become the basis for certain types of logic circuits. "That [logic ability] could change the standard paradigm of computing, by enabling computation to one day be performed in chips where data is stored, rather than on a specialized CPU," says Gilberto Medeiros Ribeiro, a senior scientist at HP Labs.



An image of a circuit with 17 memristors captured by an atomic force microscope at Hewlett-Packard's Information and Quantum Systems Lab.

IMAGE BY R. STANLEY WILLIAMS, HP SENIOR FELLOW, QUANTUM SCIENCE RESEARCH, HP LABS

The memristor has several qualities that make it attractive for memory chips. First, it is nonvolatile, so that it remembers its state after electrical current is switched off. Second, it can be scaled to a single nanometer (nm) in size, researchers believe, whereas the one-bit flash memory cell is expected to reach its scaling limit at about 20 nm. And Leon Chua, a professor of electrical engineering and computer science at the University of California, Berkeley, says the memristor's size advantage isn't its sole advantage. "You can not only build them smaller, but use fewer of them," he says. "Ten memristors might do the same thing as 50 transistors, so it's a new ball game."

In 1971, Chua published a paper, "Memristor—The Missing Circuit Element," in *IEEE Transactions on Circuit Theory*, which outlined the mathematical underpinnings of memristors, which he called the fourth fundamental building block of electronics (along with resistors, capacitors, and inductors). The existence of memristance had been reported earlier—in 1960 by Bernard Widrow at Stanford University, for example—but it was not well understood.

Earlier researchers had erroneously

interpreted memristance as a hysteresis relationship (one in which effect lags cause) between voltage and current, when in fact it is based on flux and charge, the time integrals of voltage and current, says Chua. He likens the pre-1971 view of memristance to Aristotle's belief that force is proportional to velocity and not, as Newton correctly demonstrated 2,000 years later, as proportional to the *change* in velocity, or acceleration.

In 2006, HP designed and built a titanium memristor that worked predictably and retained its state when powered off, based on the mathematical framework proposed by Chua. "For years people built [memristance] devices almost by accident. It's to the great credit of HP that they finally figured it out," he says. Figuring it out, according to HP's Stanley Williams, the chief architect of the company's memristor, meant "understanding the mathematical framework for memristors."

Almost 40 years seems a long time between the emergence of Chau's framework and the ability to reliably produce memristors, but enormous engineering hurdles had to be overcome. It required methods and tools, such as scanning tunneling microscopy, that could work at atomic scales. HP says it experimented with an enormous number of device types, many based on exotic materials and structures, but the results were often inconsistent and unexplainable. It was not until 2006 that HP developed equations that explained just what was occurring in its titanium memristors.

More Speed, Less Power

The breakthrough achieved by HP in 2006 could revolutionize memory technology, the company says. "Memristor memory chips promise to run at least 10 times faster and use 10 times less power than an equivalent flash memory chip," according to Williams, director of HP's Information and Quantum Systems Lab. "Experiments in our lab also suggest that memristor memory can be erased and written over many more times than flash memory. We believe we can create memristor ReRAM products that, at any price point, will have twice the capacity of flash memory."

The memristor could enable computation to be performed in chips where data is stored, rather than on a specialized CPU, says Gilberto Medeiros Ribeiro.

However, not everyone has been impressed by the recent announcements from HP and Rice. "The memristor is only one of several interesting [recent] flash technologies, and by no means the most interesting," says Justin Rattner, Intel's chief technology officer. "Any time someone hypes a particular memory technology before building a large memory chip, you should be suspicious, very suspicious. It's one thing to demonstrate a storage device in the lab, but it's an entirely different thing to demonstrate it can be built in high volume at low cost and with exceptional reliability."

Rattner acknowledges that flash memory, which is a $20 billion-plus market, is rapidly approaching its scaling limit. But rather than memristors, Intel is concentrating on nonvolatile phase-change memory, by which certain types of glass can be made to switch between two states by the application of heat. In the amorphous state, the atomic structure of the glass is highly disordered and has high resistivity. But when switched to its crystalline state, the glass has a regular atomic structure and low resistivity. "We built commercial-grade, phase-change memories of sufficient size to fully understand the pros and cons of the technology in a high-volume environment," Rattner says. Intel is looking at additional novel approaches to nonvolatile memories such as the spin torque transfer memory, which exploits magnetic spin states to electrically change the magnetic orientation of a material.

Society

Women and Tenure

Over the last couple of decades, women have played an increasingly important role in the research sciences. However, according to *Keeping Women in the Science Pipeline*, a recent study by University of California, Berkeley researchers, there's trouble brewing. Women, who now receive more than 50% of the Ph.D.s granted by institutions, are more likely to leave the profession than men. This, combined with growing demand for talent in Europe and Asia, puts U.S. preeminence in the sciences at risk.

The authors, who collected data from multiple sources and surveyed 62 academic institutions, found considerable differences in men and women attaining tenure track positions. Married women with young children are 35% less likely than their male counterparts to enter a tenure-track position after receiving a Ph.D. in science. What's more, married women with children are 27% less likely than men with children to receive tenure after entering a tenure-track job in the sciences.

On the other hand, single women without young children are about as successful as married men with children in attaining tenure-track jobs. According to the report, both men and women view tenure-track positions in research-intensive universities as less than a family-friendly career choice. Only 46% of the men and 29% of the women rate their institutions "somewhat" or "very" family friendly. Numerous work hours and maternity leave of less than six weeks were cited as common problems.

The upshot? The academic world needs to adopt more family-friendly policies and provide greater opportunities for female tenure-track candidates. "America's researchers do not receive enough family-responsive benefits," the report concludes. "Academia needs to be more flexible... research universities should look to build a family-friendly package of policies and resources."

—Samuel Greengard

The memristor prototype chip built at Rice is a 1-kilobyte ReRAM with sub-5 nm switches, according to Jim Tour, a synthetic organic chemist at Rice and a leading memristor researcher. Of Rattner's concern about manufacturing the devices, he says, "All so far looks good—materials cost, fabrication needs, scalability and switching times—except the switch voltage is a bit higher than we'd like, but we have some ideas to reduce that."

Rice claims to have an edge over HP's silicon-and-titanium memristor chip with its all-silicon model. "There are lots of engineering barriers to be overcome before this really takes off," says Doug Natelson, a professor of physics and astronomy at Rice. "But the use of all silicon makes the manufacturing very understandable."

Memristance comes from reduction-oxidation chemistry, in which atoms or molecules gain or lose their affinity for oxygen atoms, and in which the physical structure of materials can change. The Rice memristor chip, a thin layer of silicon oxide sandwiched between two electrodes, is made to convert back and forth between silicon (a conductor) and silicon oxide (an insulator.) A sufficiently large voltage (up to 13 volts) applied across the silicon oxide converts some of it into pure silicon nanocrystals that conduct current through the layer. The switch, according to Natelson, shows robust nonvolatile properties, a high ratio of current "on" to current "off" (>10^5), fast switching (sub-100 ns), and good endurance (10^4 write-erase cycles).

The HP version is conceptually similar, but works by the alternating oxidation and reduction of titanium. Titanium dioxide (TiO_2) is a semiconductor and is highly resistive in its pure state. However, oxygen-deficient TiO_2, which has oxygen "vacancies" where an oxygen atom would normally appear, is highly conductive. By applying a bias voltage across a thin film of semiconductor with oxygen-deficient TiO_2 on one side, the oxygen vacancies move into the pure TiO_2 on the other side of the semiconductor, thus lowering the resistance. Running current in the other direction will move the oxygen vacancies back to the other side, increasing the resistance of the TiO_2 gain.

In the short term, memristors are most likely to be used in storage devices, but eventually may be used in artificial neural networks.

In the short term, memristors are most likely to be used in storage devices, but eventually may be used in artificial neural networks, in applications such as pattern recognition or real-time analysis of the signals from sensor arrays, in a way that mimics the human brain. A memristor works like a biological synapse, with its conductance varying with experience, or with the current flowing through it over time. Similarly, the brain learns and configures itself by varying the strength of synaptic connections between neurons. The ability of memristors to remember and to work as analog devices allows them to assume any of many values over a range, just as synapses do.

The memristor self-learns from experience, and the brain is made of memristors," Chua says. "That in the long run is much more interesting and important [than data storage], because it's how you can design intelligent machines. But it's in the next 50 years, not the next 10." C

Further Reading

Chua, L.
Memristor–the missing circuit element, *IEEE Transactions on Circuit Theory 18*, 5, Sept. 1971.

Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., and Lu, W.
Nanoscale memristor device as synapse in neuromorphic systems, *Nano Letters 10*, 4, March 1, 2010.

Strukov, D.B., Snider, G.S., Stewart, D.R., and Williams, R.S.
The missing memristor found, *Nature 453*, May 1, 2008.

Tour, J.M. and He, T.
Electronics: the fourth element, *Nature 453*, May 1, 2008.

Yao, J., Sun, Z., Zhong, L., Natelson, D., and Tour, J.M.
Resistive switches and memories from silicon oxide, *Nano Letters 10*, 10, Aug. 31, 2010.

Gary Anthes is a technology writer and editor based in Arlington, VA.

Technology

Flexible Screens

Hewlett-Packard plans to deliver a prototype of a solar-powered, lightweight device with a flexible plastic screen—which HP researchers are affectionately terming "a Dick Tracy wristwatch"—to the U.S. Army later this year. Roughly the size of an index card, the low-power device will enable soldiers to read digital maps, directions, and other data on a screen that won't break or shatter like glass.

Researchers expect the HP prototype to inspire a new generation of products with flexible plastic screens, including clothing, household furnishings, and more. "You can start thinking about putting electronic displays on things where you wouldn't ordinarily think of having them," said Nicholas Colaneri, director of the Flexible Display Center at Arizona State University, in an interview with *The San Jose Mercury News*. "How about a stack of thin displays that I can peel off and stick on things, sort of like a pad of Post-it notes?"

HP hopes to produce flexible displays that can be produced continuously, like newspapers rolling off a printing press, which could be more inexpensive than the current batch production of glass displays.

The flexible display and e-reader market is expected to grow dramatically from $431 million in 2009 to $9.8 billion in 2018, according to DisplaySearch.

—Graeme Stemp-Morlock

In Memoriam | DOI:10.1145/1897852.1897862 Samuel Greengard

Gary Chapman, Technologist: 1952–2010

He raised important public issues, such as the impact of computers and the Internet on society, and encouraged social responsibility for computer professionals.

GARY CHAPMAN, AN American technologist, Internet expert, and ethicist, died Dec. 14, 2010, at the age of 58. He suffered a massive heart attack while on a kayaking trip in Guatemala.

Over the last two decades, Chapman established himself as an international authority on Internet and technology policy. He was among the first technologists to draw public attention to the issues that computing technology, including the Internet, presents to society. Chapman helped to insert ethics and human values into the world of computing by focusing on a mélange of issues, including how to address the digital divide in society, preventing the misuse of technology by government agencies, especially the military, and encouraging young people to use the Internet responsibly.

At the time of his death, Chapman was a senior lecturer at the LBJ School of Public Affairs at the University of Texas at Austin. He began teaching at the school in 1994. He also served as director of the 21st Century Project, a research and education resource for policymakers and the public.

In addition, Chapman lectured internationally and wrote for many prominent publications, including *The New York Times, Technology Review,* and *The New Republic*. His syndicated *Digital Nation* column, which appeared in more than 200 newspapers and Web sites, ran from 1995 to 2001.

Chapman's big break came in 1984 when, while a graduate student in political science at Stanford University, he learned the newly formed Computer Professionals for Social Responsibility (CPSR) was hiring its first executive director. Chapman contacted CPSR cofounder Severo Ornstein for a job interview, but was told he was too late. Ornstein had already decided on a candidate. Chapman insisted, and got an interview. "When we heard him speak, we knew he was the perfect person for the job," Ornstein recalls. "Everyone on the board agreed that we had to hire him."

PHOTOGRAPH BY SASHA HAAGENSEN

"Gary was a real pioneer in linking the lives and careers of computer professionals to the social impact of the work they do and calling for us to take responsibility for the fruits of our labors," Mitch Kapor, the founder of Lotus Development Corp., said in an email interview.

Chapman helped bring ethics and human values to the world of computing.

Chapman served as CPSR executive director from 1984 to 1991. Under Chapman's leadership, CPSR flourished and grew, and emerged as an outspoken critic of the use of technology by the military. "With Gary at the helm, CPSR raised serious questions in public forums about the Strategic Defense Initiative [Star Wars]," Ornstein says. "Gary and I shared the notion that too much technology was being developed for military purposes. We felt strongly that it would have been better if technology funding came through the National Science Foundation rather than the Department of Defense."

However, after the defeat of Star Wars and the collapse of the Berlin Wall in 1989, the threat of nuclear war diminished, and Chapman became more concerned about the effects of computers and, later, the Internet on society.

Born on Aug. 8, 1952 in Los Angeles, Chapman served as a medic in the U.S. Army Special Forces during the Vietnam War. He earned a B.A. in political science from Occidental College in 1979 and attended Stanford University's political science Ph.D. program. He left Stanford in 1984 to lead CPSR.

"Gary was deeply concerned about society plunging ahead with technology without giving adequate thought to the social implications," Ornstein says. "He helped provide much-needed direction, and he has left behind students and others who will continue to monitor and analyze technology policy." C

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

DOI:10.1145/1897852.1897863

Pamela Samuelson

Legally Speaking
Do You Own the Software You Buy?

Examining the fine print concerning your rights in your copies of purchased software.

Software companies have mass-marketed computer programs for the past few decades on terms that typically purport to restrict the right of end users to resell or otherwise transfer their interests in copies of software they have purchased. The restrictions are usually stated in documents known as shrink-wrap or click-through "licenses." Vendors of other types of digital content sometimes distribute their works with similar restrictions.

Shrink-wraps are documents inserted in packaged software, often just under the clear plastic wrap surrounding the package, informing purchasers they are not owners of copies of programs they just bought, but instead have rights in the program that are limited by the terms of a license agreement. Click-throughs are similar in substance, although the "license" terms only become manifest when you try to install the software and are directed to click "I agree" to certain terms.

Most of us ignore these documents and the restrictions they contain. Because we bought our copy of that software, we think we own it, regardless of what any "license" document says. We are also quite confident the vendor won't take any action against us, even if we do violate one of the terms, because realistically the vendor can't monitor every end user of its products.

The debate over whether mass-market transactions like these are really "sales" of goods, notwithstanding the "license" label, has been going on for decades. Strangely enough, it has yet to be definitively resolved. A recent appellate court ruling has upheld the license characterization, but a further appeal is under way in that case. This ruling is also at odds with other appellate court decisions. So things are still up in the air on the ultimate issue. This column will explain what is at stake in these battles over your rights in your copies of purchased software.

Copyright law allows rights holders to control only the first sale of a copy of a protected work to members of the public.

Three Legal Options

The distinction between "sales" and "licenses" really matters when assessing the risk of liability to copyright owners if resale restrictions are ignored.

Copyright law allows rights holders to control only the first sale of a copy of a protected work to members of the public. After the first sale, the owner of that copy is entitled to resell or otherwise transfer (for example, give it away as a gift or lend it to others) the copy free from risk of copyright liability. Bookstores and libraries are among the institutions made possible by copyright law's first-sale doctrine.

What happens if copyright owners try to restrict resales through license restrictions? There are three possible outcomes.

First, the effort to restrict resales may be deemed a nullity, as it was in a famous 1908 Supreme Court case, *Bobbs-Merrill Co. v. Straus*. Bobbs-Merrill sold books to Straus containing a prominent notice that resale of the books except at a stated price would be treated as copyright infringement. When Straus sold the books for a lower price, Bobbs-Merrill sued for infringement. The Court refused to enforce this resale restriction, saying that the copyright owner was entitled to control only the first sale of copies of its works to the public.

Second, a resale restriction may be enforceable against the purchaser insofar as he has agreed not to resell his copy, but it would be unenforceable against anyone to whom the purchaser might subsequently sell his copy.

This result might seem odd, but there is a fundamental difference between contract and property rights: Contracts only bind those who have agreed to whatever terms the contract provides; property rights create obligations that are good against the world.

A first-sale purchaser may thus have breached a contractual obligation to the copyright owner if it resells its copy of the work in violation of a resale restriction, but he is not a copyright infringer.

Those who purchase copies of copyrighted works from owners of first-sale copies are not at risk of either copyright or contract liability. These third-party purchasers are also free to resell their copies to a fourth party without fear that either is at risk of any liability to the copyright owner.

Third, courts may rule that the first-sale rule does not apply to mass market "license" transactions involving copies of copyrighted works because no "sale" has taken place. Under this interpretation, secondary markets in those copies are illegal. Anyone who purports to resell the copies is a copyright infringer for distributing copies of copyrighted works without getting permission from the copyright owner.

Vernor and *Augusto*

My March 2009 Legally Speaking column ("When is a 'License' Really a Sale?") discussed two lower-court decisions in which copyright owners challenged the resale of copies of copyrighted works on eBay. In both cases, the courts ruled that copyright's first-sale rule applied, notwithstanding transfer restrictions, because of economic realities of the transactions.

The plaintiff in *Vernor v. Autodesk* asked the court to declare that he was the owner of copies of Autodesk software he purchased from one of Autodesk's customers and that he was entitled under the first-sale doctrine to resell those copies on eBay.

Autodesk claimed no sale had taken place because the software was licensed on terms that forbade transfer of the copy to third parties. Autodesk asked the court to declare that sales of these copies on eBay constituted copyright infringement.

UMG v. Augusto involved promotional CDs of music. Augusto purchased these CDs at flea markets, online auctions, and used CD stores. Language on the CD packaging indicated they were licensed for personal use only and could not lawfully be sold or otherwise transferred to third parties. When Augusto started selling UMG promotional CDs on eBay, UMG sued him for infringement.

My March 2009 column predicted that Autodesk and UMG would appeal the trial court rulings against them, and that the software industry could be expected to push very hard for a reversal, particularly in the *Vernor* case.

The Ninth Circuit Court of Appeals heard the arguments in *Vernor* and *Augusto* on the same day. In September 2010, the appellate court ruled in favor of Autodesk. Yet it upheld Augusto's first-sale claim.

In assessing whether the first-sale rule should apply to mass-market transactions like these, it is useful to compare the economic realities test used by the trial courts in the *Vernor* and *Augusto* cases and the labeling and restrictions test adopted by the Ninth Circuit in *Vernor*.

Economic Realities Test

Under this test, a copyright owner's characterization of a transaction as a license, rather than a sale, is not dispositive. It is instead only one factor among many that should be weighed in determining the true nature of the transaction.

Other factors include whether the purchaser has the right of perpetual possession of the copy, whether the rights holder has the right and ability to reclaim the copy if the license terms are violated, whether the purchaser has paid substantial sums for the privilege of permanent possession, and whether the purchaser has the right to discard or destroy the copy. The marketing channels through which the copy was obtained (such as purchasing packages of software at Walmart or Office Depot) may also be relevant.

Under the economic realities test,

PHOTOGRAPH BY WINDELL OSKAY

Vernor and Augusto seem to be owners of copies. Those from whom they obtained the products had, it seems, the right of perpetual possession in the copies, and they could destroy or discard the copies if they wished. The software in *Vernor* had been purchased through a mass-market transaction, and the CDs in *Augusto* had been mailed for free to people who had not requested the CDs and indeed, UMG had not even kept track of the persons to whom the promotional CDs had been sent.

In a previous case, *U.S. v. Wise*, the Ninth Circuit reversed a conviction for criminal copyright infringement because the actress from whom Wise obtained a copy of a movie was the owner of that copy, notwithstanding various restrictions on what she could do with the copy, including transfers to third parties.

In Vernor's petition for rehearing by the full Ninth Circuit Court of Appeals, he argues the Ninth Circuit's ruling is in conflict with *Wise* and with precedents from other appellate courts, including *Bobbs-Merrill*.

Labeling and Restrictions Test

The Ninth Circuit in *Vernor* relied in part on *MAI Systems Corp. v. Peak Computer*, in which a Ninth Circuit panel in 1993 ruled that customers of Peak's computers, on which Peak software was installed, were not owners of copies of this software, but rather licensees. Owners of copies of copyrighted software are entitled to make copies for their use and to authorize third parties to make use-copies; non-owners are not entitled to this privilege.

MAI provided maintenance services for Peak computers to Peak customers. When MAI technicians turned on Peak computers to service them, they made temporary copies of Peak software in the random access memory. Peak argued, and the Ninth Circuit agreed, that these copies were infringing because they were not authorized by Peak.

MAI v. Peak cited no authority and offered no analysis in support of its ruling that Peak's customers were non-owners of their copies of Peak software. Peak's characterization of the transaction as a license was, for that panel, dispositive.

The three-judge panel decision in *Vernor* did not rely on the license label alone as a basis for rejecting Vernor's first-sale argument. But the license label was, as in *MAI v. Peak*, given considerable weight. The court directed that two other factors be taken into account: whether the license restricted transfers of the copies and whether it contained other substantial restrictions. The panel ruled that Autodesk should prevail against Vernor under this test. The restrictions in *Augusto*, by contrast, were less substantial than those in *Vernor*.

Software companies have been cheered by the Ninth Circuit's ruling in *Vernor*. But the rest of us should be worried about its implications.

Conclusion

Software companies have been cheered by the Ninth Circuit's ruling in *Vernor*. But the rest of us should be worried about its implications. Think about what *Vernor* may mean for flea markets, bookstores, libraries, garage sales, and auction sites. Even selling a used computer loaded with software is infringing on this theory. Think also about how easy it is for a vendor to put a "license" label on a mass-marketed product with copyrighted or patented components that states that any transfer of that copy to third parties will subject the transferor to copyright or patent infringement charges.

Consumers enjoy significant benefits from the existence of secondary markets. The first-sale limit on patent and copyrights is essential to the operation of those markets. Vernor and Augusto's cases are important to the future of competition in product markets and to preservation of the long-standing balancing principle in copyright law that the first-sale rule represents. C

Pamela Samuelson (pam@law.berkeley.edu) is the Richard M. Sherman Distinguished Professor of Law and Information at the University of California, Berkeley.

DOI:10.1145/1897852.1897864

Kenneth D. Pimple

Computing Ethics
Surrounded by Machines

A chilling scenario portends a possible future.

I PREDICT THAT in the near future a low-budget movie will become a phenomenon. It will be circulated on the Internet, shared in the millions via mobile telephones, and dominate Facebook for a full nine days. It will show ordinary people going about their everyday lives as slowly, subtly, everything starts to go wrong as described in the following events.

A middle-aged man notices the advertisements that pop up with his Web searches are no longer related to the search, but to odd and random products and services—hair replacement, sports cars, retirement homes, second career counseling, fantasy vacations, divorce lawyers, treatments for depression and impotence.

A young professional woman, recently laid off because of the bad economy, posts an Internet ad to sell her piano. The ad doesn't mention that she needs the money to pay her rent. None of the offers are as high as her asking price, but two offer exactly what she owes for rent (to the penny) and one offers exactly $100 more.

The seven most troublesome students at a high school notice that wherever they go, they run into one or more of their three meanest teachers.

An elderly couple starts hearing voices in their assisted-living apartment, faint whispers they can barely discern:

"He's awake."

"She just got out of bed."

"The coffee machine is on."

ILLUSTRATION BY VIKTOR KOEN

These merely perplexing events become ever more ominous as thousands of people, then millions, realize they are always being watched, they have no privacy, and their every decision is controlled by some unseen force. Four-sevenths of moderate Americans who are likely to vote begin to slide from the middle to the extreme right or left, not knowing why. It gets worse and worse. It seems like Armageddon.

Just as the 1956 film *Invasion of the Body Snatchers* encapsulated the Red Scare zeitgeist with its depiction of ordinary people being replaced by exact replicas who are actually aliens bent on taking over the world—as many feared that Communist spies and sympathizers were taking over America—*The Invasion of the Info Snatchers* will play on our high-tech anxiety as our online lifestyles, position-broadcasting cellphones, and protective monitoring devices are inexorably compromised, exploited, and joined by ever more subtle devices.

The preceding descriptions are intended to be satirical, but all of these scenarios are possible today, though their probability varies. What seems most unlikely to me, though, is that people are and will be nervous about being swept away in the rising tide of pervasive information technology.

Pervasive computing, ubiquitous computing, ambient intelligence, and

everyware are all commonly used terms for generally similar phenomena—the increasing presence in our everyday lives of information and communication technologies (ICT) that are too small to notice, or integrated into appliances or clothing or automobiles, or are aspects of services we use willingly. Some technologists have professed a goal for such technologies to be invisible to the users, taken-for-granted or simply unnoticed by most people, continuous with our background environment, existing and operating below and behind the realm of real-time human intervention or awareness.

These technologies were the focus of a two-day workshop held last year: Ethical Guidance for Research and Application of Pervasive and Autonomous Information Technology (PAIT). The workshop was funded by the National Science Foundation (grant number SES-0848097), with additional support from the Poynter Center for the Study of Ethics and American Institutions (http://poynter.indiana.edu) at Indiana University Bloomington, and hosted by the Association for Practical and Professional Ethics (http://www.indiana.edu/~appe). Thirty-six scholars, including ethicists, engineers, social scientists, lawyers, geographers, and social scientists, participated in the meeting, discussed ethical issues in pervasive IT, and began crafting approaches to ethical guidance for the development and use of such devices, including public policy guidance. The workshop schedule, a list of participants, and more can be found at http://poynter.indiana.edu/pait/. In this space, I cannot hope to do justice to the rich and wide-ranging conversations we had at the workshop, so I will focus on three significant topics we discussed at the workshop.

When presented properly, the benefits of pervasive IT are obvious.

Machines on the go...

When presented properly, the benefits of pervasive IT are obvious. The popularity and benefits of the Internet and cellphones need not be defended or even described, but the amount of personal information in circulation over the former is tremendous and increasing, as are the unsanctioned uses of personal data. The position-broadcasting function of the latter is not nefarious in intent. Both can be used in knowledge creation, but also for stalking of various sorts.

At the workshop, Katie Shilton, a doctoral candidate in the Department of Information Studies and a researcher at the Center for Embedded Network Sensing (CENS) at the University of California at Los Angeles, described three intriguing CENS projects using mobile phones; I'll describe two.

Participants in the Personal Environmental Impact Report (PEIR) program record and submit a continuous location trace using their mobile devices. A participant's location is captured every few seconds, allowing the system to determine the location and infer the most likely mode of locomotion—foot, car, or bus. The participant's travel profile is then correlated with Southern California air quality and weather data, allowing PEIR to estimate the participant's carbon footprint, as well as her or his exposure to air pollution. The accuracy of the data gives an unprecedented look into the environmental harms people create and suffer.

Through the Biketastic project, bicyclists carrying GPS-enabled mobile phones transmit data on their routes through L.A. The information is not limited to position, but also includes data on the volume of noise and, using the device's accelerometer, the roughness of the path. The data is transmitted to a Web site (http://biketastic.com) and displayed. Future improvements could display existing data about the route, such as air quality, traffic conditions, and traffic accidents. The Biketastic riders can also share their information with other riders to create a detailed profile of the rideability of numerous routes.

Shilton described not only the benefits and uses of these projects, but the potential down-side and abuses. Summarizing the latter, she asked, "How do we build and shape such pervasive sensing systems without slipping into coercion, surveillance, and control?"

...in the home...

Kalpana Shankar, an assistant professor in the School of Informatics and Computer Science and an Adjunct Assistant Professor in the School of Library and Information Science at Indiana University Bloomington, distributed a case study to participants before the workshop. The case study, "Sensing Presence and Privacy: The Presence Clock," was developed as part of an NSF-funded research project, "Ethical Technology in the Homes of Seniors," or ETHOS (http://ethos.indiana.edu/).

An increasing number of people want to live in their own homes as they age and begin to become less self-reliant due to slowly increasing frailty of various sorts. Their offspring want to ensure they are safe and that responses to mishaps are rapid and certain. The ETHOS project investigates how Internet-connected devices that alert responsible parties to troubling changes in routine—such as a person falling in the living room and not getting up—can give care providers peace of mind and elders more autonomy than they would enjoy in an assisted-living facility as well as life-saving interventions in an ethical manner acceptable to both elders and their offspring.

The Presence Clock case can be found at the PAIT blog (http://ethicalpait.blogspot.com/2009/08/case-study-presence-clock.html) along with commentary. Briefly described, the Presence Clock is an analog clock that comes in pairs. One clock is installed in the elders' living space and the second in the living space of their caretakers. The two clocks are connected via the Internet. Both clocks sense movement and presence and lights on the remote clock show roughly how much time someone spent at any given hour in the room with the local clock; the time spent is indicated by the brightness of a light next to the relevant hour marker (for example, a dull light at 1 and a bright light at 4 indicate someone spent little time near the clock at 1 and a good deal of time there at 4). A different-colored light blinks next to the hour marker when someone most recently entered the room.

The goal of the Presence Clock is to give the elder and caregiver a sense of mutual presence, even at a distance. A glance at the clock can give either party a sense of what the other has been doing; a change in routine might prompt a telephone call. It is less intrusive than a true surveillance system with an audio or visual feed, but could afford a great deal of comfort to both parties and enable an elder to stay in her or his own home longer than the caretakers would otherwise feel comfortable.

But it could feel intrusive to the elder, a trade-off between privacy and security that he or she does not want to make, but the caretaker insists upon. Responsible development and deployment of the Presence Clock is not just a technical and marketing challenge, but also a challenge in human relations and customer/user education.

...and thinking for themselves.

Perhaps even more troubling than machines that hide or drop from our awareness are machines that make choices without direct human intervention—generally termed "autonomous systems." Keith W. Miller, the Louise Hartman Schewe and Karl Schewe Professor of Computer Science at the University of Illinois at Springfield (to whom I owe the title of this column) highlighted one concern about autonomous systems in a presentation called "The problem of many hands when some of the hands are robotic," in which he revisited Helen Nissenbaum's 1994 article, "Computing and accountability."[1] The problem of many hands lies in discerning or assigning responsibility when something goes wrong. The more people involved in a project, the more people there are to whom the blame can be shifted. When the technology is believed to be able to learn on its own and make its own decisions, there can arise a temptation—perhaps even a compulsion—to blame the machine itself, allowing the humans involved in its design, construction, and deployment to wash their hands of it.

I think it's fair to say that most of the workshop participants deplored this tendency. Determining moral responsibility is a serious endeavor, and dodging or shifting blame (if that's all one does) is irresponsible in itself. At the workshop Miller started advocating for an effort to make a strong statement about the importance of accepting moral responsibility even in circumstances of complicated causality. Our time was too short to make much progress, but Miller has pushed the project forward in the interim. As I write this column in early 2011, Miller is working on the 27th draft of "Moral Responsibility for Computing Artifacts: Five Rules" (The Rules, for short) and has assembled a 50-member, international Ad Hoc Committee for Responsible Computing to improve drafts. It's remarkable to get such cooperation and consensus from scholars solely over email; more so is that the document is only four pages long (see https://edocs.uis.edu/kmill2/www/TheRules/).

Determining moral responsibility is a serious endeavor, and dodging or shifting blame (if that's all one does) is irresponsible in itself.

Conclusion

I have only touched on what happened at the workshop itself, and mentioned only one of the ongoing projects the workshop inspired. More is happening and still more will be accomplished thanks to the enthusiasm of this remarkable group of scholars, the ripple effect that will let this workshop touch many more people than those who attended, and a small grant from the National Science Foundation. C

Reference

1. Nissenbaum, H. Computing and accountability. *Commun. ACM 37*, 1 (Jan. 1994), 72–80.

Kenneth D. Pimple (pimple@indiana.edu) is Director of Teaching Research Ethics Programs at the Poynter Center for the Study of Ethics and American Institutions, an endowed center at Indiana University-Bloomington, and on the Affiliate Faculty of the Indiana University Center for Bioethics.

Calendar of Events

March 19–23
Computer Supported Cooperative Work,
Hangzhou, China,
Sponsored: SIGCHI,
Contact: John Tang,
Email: johntang@microsoft.com

March 21–25
Tenth International Conference on Aspect-Oriented Software Development,
Porto de Galinhas, Brazil,
Contact: Borba Paulo,
Email: phmb@cin.ufpe.br

March 22–24
6th ACM Symposium on Information, Computer and Communications Security,
Hong Kong,
Contact: Lucas Chi Kwong Hui,
Email: hui@cs.hku.hk

March 22–24
4th International ICST Conference on Simulation Tools and Techniques,
Barcelona, Spain,
Contact: Liu Jason,
Email: liux@cis.fiu.edu

March 28–29
International Cross-Disciplinary Conference on Web Accessibility,
Andhra Pradesh, India,
Contact: Ashley Cozzi,
Email: cozzi@hq.acm.org

March 30–April 1
8th USENIX Symposium on Networked Systems Design and Implementation,
Boston, MA,
Contact: David G. Andersen,
Email: dga@cs.cmu.edu

April 1–2
Consortium for Computing Sciences and Colleges (CCSC) Midsouth,
Conway, AR,
Contact: Larry Morell ,
Email: lmorell@atu.edu

April 1–2
Consortium for Computing Sciences and Colleges (CCSC) Southwestern,
Los Angeles, CA,
Contact: Stephanie August,
Email: saugust@lmu.edu

DOI:10.1145/1897852.1897865

Peter J. Denning

The Profession of IT
Managing Time

Professionals overwhelmed with information glut can find hope from new insights about time management.

Time management is a persistently hot issue for many computing professionals. Almost every day we hear (or have) laments about information overload, about a relentlessly increasing rate of input from Internet and other sources, and about feelings of overwhelm, data drowning, inadequacy, and even victimization. The consequences from poor time management can be significant: loss of trust, loss of reputation, negative assessments about our competence and sincerity, and inability to get the jobs and projects we want. Books and seminars on time management continue to be popular. Software tools to help keep track of calendars and to-do lists sell well.

The same issues plague us as decision makers. We wanted larger networks and more sensors for better situational awareness—and now those networks overwhelm us. We still complain about the quality of our decisions.

In my own research on this subject I have turned up new insights that are very helpful especially if viewed as a coherent framework. I discuss these insights here. There are opportunities here for all computing professionals to become more productive and for some to design new software tools.

From Time Management to Commitment Management

It is very important to frame the question properly. Although we often complain about not having enough time, lack of time is the symptom, not the problem. The problem is commitment management. Time is one of the resources needed to manage commitments. Other resources, such as money, space, and personnel, may be needed as well. From now on, let us talk about commitment management.

Although we often complain about not having enough time, lack of time is the symptom, not the problem.

In managing commitments we need to know only four things. I'll call them *practices* because you can learn them as skills and get tools to help you do them better (see the figure here).

1. How to track commitments to their completions;
2. How to chose what commitments to make or decline;
3. How to organize the conversations that lead to completions of commitments; and
4. How to manage mood and capacity.

These four practices go together. If we pay attention to only one, we will see some headway but not a lasting solution to our problem.

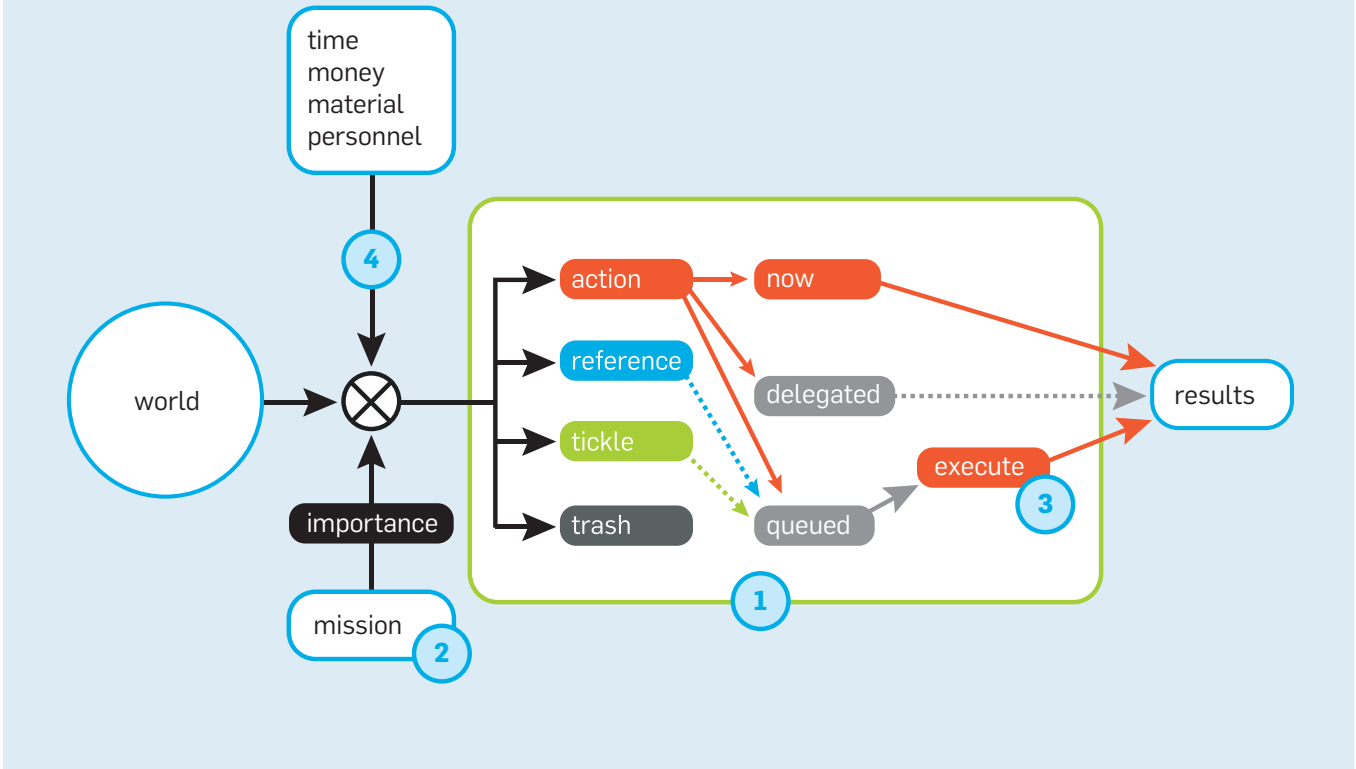
Tracking Commitments to Completion

Much of the literature on time management focuses on the first practice. That practice directly addresses one of the biggest breakdowns with commitment management—missed or forgotten commitments. When the world gets demanding, we can find ourselves in a state of constant worry about whether we forgot commitments or their due dates and whether we have the capacity to get everything done.

David Allen has written a hugely popular book about how to organize our records so that nothing is lost and we can eliminate from our minds all concerns about whether every commitment is being taken care of.[1] He has defined an operating system for managing commitments. His system can be implemented with a few simple rules and folders. The folders and structure of flows among them are remarkably similar to the job-scheduling part of a computer operating system. After you set up your system and practice its rules for a short time, you soon become skilled at commitment tracking. That so many people have found his book really helpful illustrates that the record-keeping part of commitment tracking is a huge struggle for many.

Allen's story begins with "stuff" arriving before you. Stuff is anything that demands your attention and possible future action. Think of stuff as incoming requests. A request can be anything from the really simple (such as "read me" or "take note") to the complex (such as "write an analytic report" or

Figure 1. Four practices of commitment management: 1-tracking, 2-selecting, 3-executing, 4-capacity planning.



"implement a software tool"). Allen says to sort the incoming items into trash (ignore and delete), possibly useful (save in tickler file), reference (save in reference file), and actionable. You do actionable items immediately if they require two minutes or less (for example, a quick answer to an email message); otherwise you enqueue them in your to-do list and calendar, or you delegate them. You review your queues periodically to see if your delegations have completed and the orderings of lists reflects your current priorities. Once an item is in this system, you do not have to think about it and your mind is clear to focus on the tasks needing completion.

This story is incomplete in three ways. (1) It does not address the possibility of controlling the flow of stuff. (2) It does not make explicit that much of the stuff originates with you and your teams as you design actions to fulfill your own commitments. And (3), it does not deal with limitations on your capacity and the mood of overwhelm when you are beyond capacity. These three aspects take us to the next three practices.

Trump the Urgent With the Important

Stephen Covey has discussed at length the notion of controlling what commitments you enter or decline.[2] The central question is: what exactly do you commit to? Covey maintains that the answers come from having a clear sense of mission. Just as organizations have mission statements, individuals should have personal mission statements. We can ignore requests that do not serve our mission, and we can (politely) ask the people making them to leave us alone. Covey counsels each of us to write down a mission statement, including our ongoing personal and professional commitments. Then we arrange our calendars to make sure that we allocate time sufficient for each major commitment.

Covey argues that good mission statements help people distinguish important requests from urgent requests. Many people find themselves overwhelmed with urgent but unimportant requests that consume all their time. This is a double whammy—they are frustrated at being unable to find time for the important things and exasperated over the sheer number of urgent, time-wasting requests. The irony is that many urgent requests are the result of previously neglected important tasks. For example, if you make sure you give excellent service to your customers, you will not spend a lot of time answering complaints.

Covey tells an engaging story about a time-management seminar leader who did a demonstration involving placing rocks, then gravel, sand, and water into a large glass jar. After his students struggled with getting all these items successfully into the jar, he asked, "What is the point about time management?" He got many answers including there is always more room to fit more things in your schedule if they are small or liquid enough, and you may therefore have more capacity to get things done than you think. He said, "No. The point is that if you don't put the big rocks in at the beginning, you can't get them in at all."

The moral for commitment management is: let your mission statement inform you about what tasks are most important, then set aside sufficient time in your schedule to do them.

Mastering Conversations for Context, Possibility, Action

The third practice begins with the realization that all commitments are made in conversations.[3] The practice is to become an observer and facilitator of those conversations. There are three basic kinds of conversations.

- *Context*. Define the purpose, meaning, and value of actions.
- *Possibility*. Invent possibilities for future action (in the context).
- *Action*. Elicit the commitments that will realize specific possibilities and see them through to completion.

It would be a misunderstanding of Allen's model (practice 1) to interpret his "actionable" items only in the third sense. Professionals who do not create context will find it difficult to get anyone to work with them. Although the action itself is performed in the third conversation, the other two are needed before people are willing to engage in a conversation for action. Sometimes you need to schedule time for context and possibility conversations, but more often you can insert them as needed as prefaces to your requests and offers (which open conversations for action).

A conversation for action takes place between a customer and performer; the customer makes a request (or accepts an offer) that the performer commits to fulfilling. The transaction between them can be visualized as a closed loop with four segments: request, negotiate, perform, accept.[5] Performers often make requests of others to get components for their own deliveries; thus a single request can evoke coordination in a larger network of people (for details on conversations for action and their skilled management, see [3–5]).

Commitment management presents a big software challenge.

To manage commitments means to manage the conversations leading to the fulfillment of those commitments. Have you or someone made the appropriate requests or offers? Who is responsible for performing each action? Who is responsible for accepting and declaring satisfaction with the result? Do you trust promises made to you by others along the way?

Managing Capacity and Mood

The final aspect of the picture is your ability to manage your capacity and mood. You have the capacity for a commitment if you have the time and other resources needed to fulfill the commitment. If you do not have the resources, you will need to initiate conversations to get them—and you must manage those conversations as well. Generally, if you have accepted too many commitments relative to your capacity, you will feel overwhelmed, victimized, and sometimes panicked—poor moods for productivity. When you do not have the capacity, you can find yourself in a death spiral of an increasing backlog of broken promises, negative assessments about your performance, lack of willingness to trust you, and a personal sense of powerlessness. Over time, these bad moods increase stress and anxiety in your body and lead to chronic diseases. Not a pretty picture.

With a simple exercise, you can assess whether you have the capacity for your commitments and take corrective steps when they are beyond your capacity.[3,4] On a three-column spreadsheet, make one row for each commitment. Put a description of the commitment in the first column, the number of weekly hours you need to do it well in the second column, and the number of weekly hours you actually spend in the third. Make sure to include all your "big rock" commitments including time for family, sleep, and exercise. Many people who feel chronically overwhelmed discover from the exercise that their column-two total exceeds 168, the number of hours in a week. Even if the column-two total fits, they discover that their column-three total exceeds 100 hours per week for professional commitments. In contrast, people who feel productive and satisfied usually do not spend more than 60–80 hours per week on professional commitments.

You need to reduce your load if you are over capacity. First, look at your mission statement and recall what is most important to you. Make sure that the time you allocate for your "big rock" commitments is sufficient to do them right. All other commitments need to be modified or eliminated. Modified means you negotiate new terms with the person(s) who expects the results. Eliminated means you cancel the commitment. In both those cases you need to work with the customers of your commitments to reset their expectations and take care of any consequences resulting from your scale-back or cancellation.

Conclusion

Commitment management presents a big software challenge. There are software tools that help with some of the four practices separately. For example, *OmniFocus* (omnigroup.com), *Things* (culturedcode.com), and *Taskwarrior* (taskwarrior.org) conform to Allen's workflows in practice 1. *Orchestrator* (orchmail.com) tracks conversations for action through their stages in practice 3; *ActionWorks* (actiontech.com) goes further, mapping and managing entire business processes. Can someone design a coherent system that supports all four together?

If you learn the four commitment-management practices, you will be able to execute all your commitments productively and in a mood of fulfillment and satisfaction. All your customers will be satisfied and you will enjoy a strong, trustworthy reputation. C

References

1. Allen, D. *Getting Things Done: The Art of Stress-Free Productivity*. Penguin, 2001.
2. Covey, S.R., Merrill, R., and Merrill, R. *First Things First*. Simon & Schuster, 1994.
3. Denning, P. and Dunham. R. *The Innovator's Way: Essential Practices for Successful Innovation*. MIT Press, 2010.
4. Denning, P. Accomplishment. *Commun. ACM 46*, 7 (July 2003), 19–23. DOI= http://doi.acm.org/10.1145/792704.792722.
5. Winograd, T. and Flores, F. *Understanding Computers and Cognition*. Addison-Wesley, 1987.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for innovation and information superiority at the Naval Postgraduate School in Monterey, CA and is a past president of ACM.

DOI:10.1145/1897852.1897866

Daryl E. Chubin and Roosevelt Y. Johnson

Broadening Participation

A Program Greater than the Sum of Its Parts: The BPC Alliances

Changing the trajectory of participation in computing for students at various stages of development.

There is virtually no discipline or aspect of our daily lives that is not positively impacted by advances in computer science. It has become the backbone of our technologically dependent society. In fact, computer software engineers are among the occupations projected to grow the fastest and add the most new jobs over the 2008–2018 decade.[6] Yet, bachelor's, master's, and Ph.D. degrees earned by U.S. citizens and permanent residents continue to decline.[3,7] Further, degrees earned by women, persons with disabilities, and underrepresented minorities (American Indians/Alaskan Natives, African Americans, Native Hawaiian's/Pacific Islanders or Hispanics) lag those of non-resident Aliens, Asians, and White males.

Students at the CAHSI 2009 annual meeting held at Google headquarters.

PHOTO COURTESY OF AAAS CENTER FOR ADVANCING SCIENCE & ENGINEERING CAPACITY

Program Focus

Rather than focus on the problems that beset computing, we will emphasize solutions in the form of the National Science Foundation's (NSF) Broadening Participation in Computing (BPC) program.[a] The BPC-A program supports three categories of awards: Alliances; Demonstration projects (DPs); and Leveraging, Scaling, or Adapting Projects, or Demonstration Projects (LSA). Typical DPs pilot innovative programs that, once fully developed, could be incorporated into the activities of an Alliance or otherwise scaled for wider impact. LSA projects can leverage, scale, and adapt the work of Alliances or DPs, as well as efforts by other organizations to extend the impact of effective practices. Alliance and Alliances Extension Projects (Alliances) represent broad coalitions of academic institutions of higher learning, secondary and middle schools, government, industry, professional societies, and other not-for-profit organizations designing and carrying out comprehensive programs to reduce underrepresentation in the computing disciplines. Projects may target stages of the academic pipeline from middle school through the early faculty ranks, and are expected to have significant impact on both the quality of opportunities afforded to participants and the number of participants potentially served.[5]

NSF funding for the Alliances began in 2005/2006 with most programs operating with students approximately one year later. Ten alliances constitute the core of BPC as of 2009. An eleventh alliance, the National Center for Women & IT (NCWIT), predated the BPC program, but has served as a focal point and resource for all the Alliances, par-

a For additional information on the BPC program, visit: http://www.bpcportal.org/bpc/shared/home.jhtml.

NSF BPC Alliances*

A4RC (N. Carolina A&T)	www.a4rc.org
AccessComputing (U. Washington)	www.washington.edu/accesscomputing
CAHSI (U.Texas-El Paso)	http://cahsi.org
Computing Research Association-Women/Coalition to Diversify Computing	www.cra-w.org and cdc-computing.org
STARS (U. N. Carolina)	www.starsalliance.org
ARTSI (Spelman College)	http://artsialliance.org/
CAITE (U. Massachusetts)	http://caite.cs.umass.edu
El (Rice U)	http://empoweringleadership.org
GeorgiaComputes! (Georgia Tech)	http://gacomputes.cc.gatech.edu/
Into the Loop (U. California, LA)	http://intotheloop.gseis.ucla.edu
NCWIT (Nat'l Ctr for Women & IT)	http://www.ncwit.org

*as of 2009

ticularly for those focusing on gender. Jointly, these 11 Alliances have blanketed computer science with alternatives for diversifying participation in computer careers (see the table here).

In unison, the BPC Alliances endeavor to significantly increase the number of U.S. citizens and permanent residents receiving degrees in the computing disciplines, with an emphasis on students from communities long underserved in computing. Cohorts of students—those steeped in poverty, first-generation college-goers, ignored through stereotyping and low expectations, academically underprepared or not resembling what computer scientists traditionally look like—are being reached, gaining confidence and skills, and making progress toward degrees and careers in computing.

By working to create a critical mass on campuses, the BPC Alliances have built their own infrastructure(s), encompassing both the physical (facilities, instrumentation) and the social (networks, partnerships) components of their activities. The Alliances extend organizational commitments to educate, train, and utilize science, technology, engineering, and mathematics (STEM) professionals in various communities. Individually impressive, the BPC Alliances are more than the sum of their parts and greater than the sum of their experiments. Some Alliances target specific races or ethnicities, others direct their efforts toward women or persons with disabilities, and several Alliances reach to all underrepresented groups. Together, the Alliances are a cohesive entity, providing the field with alternatives that enable participation by all.

The BPC Alliances are not about quick fixes. Rather, they aim to produce systemic changes—changing the trajectory of participation in computing for students at various stages of development. Such change comes not only through impacts on individual students and educators, but also as institutions adjust their approaches and structures to enhance the teaching and learning of computing. The Alliances embody goal-directed change across the educational spectrum. Simply put, they develop talent. Students are encouraged to work hard and be successful, whether that means entering the work force upon high school graduation or pursuing a college or advanced degree.

The American Association for the Advancement of Science (AAAS) Center for Advancing Science & Engineering Capacity[b] staff conducted a three-year portfolio assessment of the Alliance component of the BPC program. The Capacity Center found the 11 BPC Alliances are implementing various methods to attract, nurture, and retain students, using innovative practices and strategies.[2] Four approaches are demonstrating success:

▸ *Reforming statewide systems*. Alliances work across different institutions and systems (for example, K–12, two-year, and four-year) to develop a common data framework enabling them to focus on students and educational systems as well as operate on various levels of the education pathway.

▸ *Focusing on undergraduates*. Alliances employ varied methods, such as introductory computer classes designed to attract majors and bolster underprepared students; peer-facilitation in the gatekeeper courses; undergraduate professional socialization and research experiences; mentoring; developing undergraduates' technical excellence, leadership skills, and civic engagement around computing; and, partnering undergraduates with younger students so that both are motivated to reach their personal best in computing.

▸ *Connecting unlike institutions/Creating new partnership models*. Alliances build productive relationships between dissimilar institutions (for example, the University of California, Los Angeles with the Los Angeles Unified School District; Historically Black Colleges and Universities with top research universities). These models feature novel research collaborations, team learning, and multiple educational pathways.

▸ *Creating national, interlocking networks*. Alliances socialize computer science students at all levels, providing students and educators with opportunities to share experiences and develop professional skills and knowledge.

Regardless of approach, the Alliances are committed to collaboration, serving on each others' boards, conducting face-to-face meetings of senior personnel, contributing to the BPC Portal, and disseminating results. Virtually all of the Alliances work with AccessComputing to increase their inclusion of persons with disabilities, send students and faculty to the annual STARS Celebration, and encourage their students to join the Empowering Leadership (EL) Alliance and participate in the CRA/CDC's programs. Together, they are forming a national infrastructure for change. This includes devising a common core of indicators to measure and monitor Alliance progress (see BPC Common Core Indicators[1]).

A Snapshot of the Alliances

Each BPC Alliance has a storyline that conveys the excitement of its work. We encourage readers to visit the Alliance

b The AAAS Center for Advancing Science & Engineering Capacity is a fee-for-service consulting group that provides institutions of higher education with assistance in improving delivery of their educational mission, especially in science, technology, engineering and mathematics fields. Details can be found at www.aaascapacity.org.

Web sites listed in the accompanying table to obtain in-depth and updated information.

While seeking to increase African-Americans' entry into computing research careers, the Alliance for the Advancement of African American Researchers in Computing (A^{4}RC) and the Advancing Robotics Technology for Societal Impact (ARTSI) connect students at Historically Black Colleges and Universities with the resources of top research institutions. A^{4}RC covers a range of research topics; ARTSI focuses entirely on robotics.

The AccessComputing Alliance strives to increase the number of students with disabilities who complete postsecondary computing degrees and enter the computing work force. The program leads capacity-building institutes for computing departments. The Computing Alliance of Hispanic-Serving Institutions' (CAHSI) interventions center on undergraduates and the gateway of introductory courses, as well as the power of peer groups, to increase the number of Hispanic students entering the computing professoriate and work force.

With a goal of increasing women's participation in information technology, the NCWIT has programs in K–12 education, college-level outreach and curriculum reform, corporate recruitment and retention, and entrepreneurial endeavors.

Other Alliances work to increase the numbers of all minorities. For example, Students & Technology in Academia, Research & Service (STARS) targets undergraduates and directs its efforts toward all underrepresented groups, including those with disabilities. Its centerpiece is the STARS Leadership Corps, a program that draws students from all member institutions in yearlong, team-based leadership projects. Also working to increase the number of minorities is the EL Alliance, a program that provides a safety net to its participants by fostering networking opportunities, ongoing communication, and a shared learning experience.

Using community colleges as its centerpiece, the Commonwealth Alliance for Information Technology Education (CAITE) focuses on women and minorities in groups that are underrepresented in the Massachusetts innovation economy, that is, economically, academically, and socially disadvantaged students. Georgia Computes! works to attract women and minorities into computing by building a computing education pipeline across the state of Georgia. Into the Loop aims to increase the computer science learning opportunities of students in the Los Angeles Unified School District and broaden the participation of African Americans, Hispanics, and girls in computing via the Computing Science Equity Alliance.

The BPC Alliances have made the field of computing more real and more attainable for many students.

Looking Ahead

The BPC Alliances have made the field of computing more real and more attainable for many students. In accomplishing this feat, BPC Alliances highlight at least two key characteristics of good alliances: the ability to collaboratively adjust approaches, structures, and practices; and the ability to develop new communication infrastructures to more effectively plan, implement, evaluate, and broadly disseminate effective practices.

In FY 2011, the NSF's Division of Computer and Network Systems is investing in a comprehensive Education and Work Force (EWF) Program. The BPC Alliance Program, Computing Education for the 21st Century (CE21), and the Graduate Research Fellowship Program will be funded as part of that activity. The Computing Education for the 21st Century (CE21) program seeks to increase competencies for *all* students, regardless of gender, race, ethnicity, disability status, or socioeconomic status. By promoting and enhancing K–14 computing education, it will enhance interest in and student preparation for careers in computing-intensive fields. CE21 will support Type I (smaller-scale studies of the effectiveness of new instructional materials and interventions and strategies to develop K–14 teaching expertise), Type II (proven effective implementations taken to scale), and Planning proposals (support for the establishment of new partnerships and collaborations to develop Type I and Type II proposals)[5].

Through the EWF program, NSF seeks to build on the foundation of the BPC Alliances to reach more students and illuminate pathways into computing. An important stated goal of the EWF program, in fact, is to "transform computing education at all levels and on a national scale to meet the opportunities of a world where computing is increasingly essential to all sectors of society." Considering the multiple stakeholders involved, the importance of cultivating interpersonal relationships, forging and embracing shared values, and using process and outcome data to monitor and evaluate Alliance contributions to computing, the BPC Alliances are fulfilling the expectation of how transformative models of intervention in a STEM discipline look and function. C

References

1. BPC Common Core Indicators, Post-Workshop Version. Washington, DC: AAAS Working Paper, Feb. 9, 2010; http://php.aaas.org/programs/centers/capacity/07_Engagement/07_BPCProgram.php.
2. Chubin, D.E. and Johnson, R.Y. Telling the Stories of the BPC Alliances: How One NSF Program Is Changing the Face of Computing. AAAS, Washington, D.C., June 2010; http://php.aaas.org/programs/centers/capacity/documents/BPC%20Stories.pdf.
3. *Computing Research News*. Computing Research Association, Washington, D.C., 2010; http://www.cra.org/uploads/documents/resources/taulbee/0809.pdf.
4. Congressional Testimony, Oct. 6 1999. Computing Research Association, Washington, D.C., 2004; http://archive.cra.org/Policy/testimony/lazowska-5.html.
5. National Science Foundation: Broadening Participating in Computing Program. National Science Foundation, Arlington, VA; http://www.bpcportal.org/bpc/shared/about.jhtml.
6. U.S. Bureau of Labor Statistics, Office of Occupational Statistics and Employment Projections. Government Printing Office, Washington, D.C., 2009; http://www.bls.gov/oco/ocos303.htm.
7. WebCASPAR. National Science Foundation, Arlington, VA, 2010; https://webcaspar.nsf.gov/.

Daryl E. Chubin (dchubin@aaas.org) is the director of the Center for Advancing Science & Engineering Capacity at the American Association for the Advancement of Science in Washington, D.C.

Roosevelt Y. Johnson (rjohnson@aaas.org) at the time of the work reported here, was a Fellow at the Center for Advancing Science & Engineering Capacity at the American Association for the Advancement of Science in Washington, D.C., on leave from the National Science Foundation.

DOI:10.1145/1897852.1897867

Marc Snir

Viewpoint

Computer and Information Science and Engineering: One Discipline, Many Specialties

Mathematics is no longer the only foundation for computing and information research and education in academia.

During the last 60 years we have seen the beginning of a major technological revolution, the *Information Revolution*. IT has spanned large new economic sectors and has, over a long period, doubled the rate of increase in labor productivity in the U.S.[1,16] Over two-thirds of job openings in science and engineering in the coming decade are in IT.[12] Intellectual property, rather than physical assets, has become the main means of production: control over intangibles (such as patents and copyrights) are at the forefront of the national and international business agenda;[6,23] investment by industry in intangible assets has overtaken investment in tangible means of production.[7,19]

The information revolution is far from having run its course: "machine-thought" has not yet replaced "brain-thought," to the extent that "machine-made" has replaced "hand-made." One can be confident that the use of digital technologies will continue to spread; that more and more workers will move from the physical economy to the information economy; and that people will spend more and more of their work and leisure time creating, manipulating, and communicating information.

The fast evolution of IT motivates a periodic reexamination and reorganization of computing and information (C&I) research and education in academia. We seem to be in one such period. Many universities have established or expanded schools and programs that integrate a broad range of subdisciplines in C&I; and NSF is affecting the scope of research and education in C&I through the creation of programs such as the Cyber-Enabled Discovery and Innovation (CDI) and Pathways to Revitalized Undergraduate Computing Education (CPATH) programs.[21,22]

The fast evolution of IT motivates a periodic reexamination and reorganization of computing and information (C&I) research and education in academia.

I strongly believe that C&I is one broad discipline, with strong interactions between its various components. A coherent view of the whole must precede any discussion of the best ways of dividing it into subdisciplines. The dominant discourse in our community should be about building a coherent view of the broad discipline, building bridges between its constituents, and building bridges to other disciplines as we engage in interdisciplinary research. I hope this column will contribute to these goals.

C&I Is a Use-Driven Research Discipline

I am discussing in this column the broad field of Computing and Information Science and Engineering (CISE): the study of the design and use of digital systems that support storing, processing, accessing and communicating information. To prevent possibly misleading connotations, I shall call this broad field *Computing and Information* (C&I).

We still seem to be debating whether computer science is science, engineering, or something unlike any other academic discipline (see, for example[9,11]). The debate is often rooted in a linear view of science and engineering: Scientists seek knowledge, for knowledge sake; through a mysterious process, this knowledge turns out to have practical consequences and is picked up by applied scientists, next engineers, and then used to develop better technologies. This view encourages an implicit value system whereby science is seen a higher call than engineering.

Donald Stokes, in his book *Pasteur's Quadrant*,[25] leads a powerful attack against this simplistic view of science. He points out that, over the centuries, fundamental research has been often motivated by considerations of use—by the desire to implement certain processes and achieve certain goals—not (not only) by the desire to acquire knowledge for knowledge's sake. His paradigmatic example is Pasteur, who founded modern microbiology, driven by the practical goal of preserving food.

According to Stokes, research should be described as a two-dimensional space, as shown in Figure 1. Stokes further argues that "Pasteur's Quadrant," namely use-inspired basic research, is increasingly prevalent in modern research institutes. The argument of Stokes strongly resonates with schools of engineering, or computer science. Most of their faculty members pursue scientific research that has a utilitarian justification; their research is in "Pasteur's Quadrant."

Any engineering department in a modern research university is a science and engineering department. This is often indicated by the department's name: Material Science and Engineering, Nuclear Science and Engineering, or even Engineering Science (at Oxford University). Figure 2 describes the research activities in such a department.

Faculty members perform basic use-inspired or applied research related to the applications of their discipline. The foundations guiding this research and constraining the engineering design space are natural sciences—mostly physics.[a] The practical goal of their research is to enable the production of better artifacts or better processes. The design of and experimentation with prototypes often is an essential step in the transfer of knowledge from research to practice, as they provide a proof of concept, a test and validation for theories, and a platform to experiment with design alternatives. *I believe it is the richness of the feedback loops between research and practice and between basic research and applied research that best characterizes top engineering departments.*

a Using the definition of engineering as "design under constraints."[28]

The diagram in Figure 2 describes not only engineering departments, but also other use-oriented disciplines such as medicine or agriculture. Furthermore, concern about impact and use, and research in "Pasteur's Quadrant," are increasingly prevalent in science departments, be it life sciences, social sciences, or physical sciences. Only a few purists would claim that departments are weakened by such concerns.

The diagram in Figure 2 clearly applies to C&I. Our discipline is use-inspired: We want to build better computing, communication, and information systems. This occasionally motivates use-inspired basic research (for example, complexity, cryptography), and often involves applied research (such as architecture, databases, graphics). The design and experimentation with prototypes is essential in system research. C&I scientists use scientific methods in their research;[8,10] and there is a continued back and forth between basic and applied research and between academic research and the development of digital products and services by industry.

C&I Needs Broader Foundations

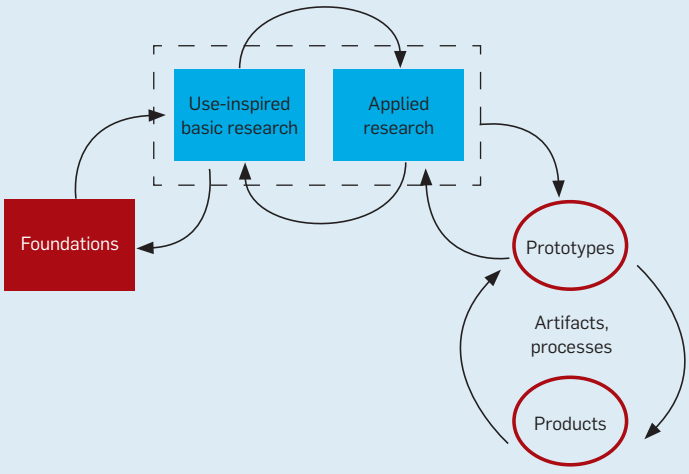
C&I was lucky to develop early on mathematical abstractions that represented important constraints on computing devices, such as time and space complexity; this enabled C&I to develop useful artifacts while being fully contained within the confines of mathematics: The early development of algorithms, programming languages, compilers or operating systems required no knowledge beyond C&I and its mathematical foundations.

Mathematics continues to be the most important foundation for C&I: The artifacts produced by C&I researchers and practitioners are algorithms, programs, protocols, and schemes for organizing information; these are mathematical or logical objects, not physical objects. Algorithms, programs or protocols are useful once realized, executed or embodied in a physical digital device; but they are mostly studied as mathematical objects and the properties studied do not depend on their physical embodiment. Indeed, one might call much of C&I "mathemati-

Figure 1. Pasteur's Quadrant (adapted from Stokes[28]).

		Consideration of use?	
		No	***Yes***
Quest for fundamental understanding?	***No***		Pure applied research (Edison)
	Yes	Pure basic research (Bohr)	Use-inspired basic research (Pasteur)

Figure 2. Modern engineering research.



cal engineering"[b]: it is focused on the creation of new mathematical objects under constraints, such as low time and space complexity for discrete algorithms, good numerical convergence for numerical algorithms, or good precision and recall for classifiers; the difference between mathematics and "mathematical engineering" is precisely the emphasis on such constraints.

As technology progresses, new constraints need to be considered. For example, time complexity is increasingly irrelevant when communication (to memory, disk, and network) replaces computation as the main performance bottleneck, and when energy consumption becomes the critical constraint. New technologies that will take us "Beyond Moore's Law" (quantum computing, molecular computing) will require new mathematical abstractions.

Part of C&I, namely computer engineering, has always been concerned with the interplay between the mathematical abstractions and their physical embodiment. In addition to mathematics, physics is foundational for this specialty, and will continue to be so. Physics is also important for cyber-physical systems that directly interact with their physical environment.

b Mathematical engineering was apparently used as a synonym for "computer science" in Holland, in the early days of the discipline. It is now used by some schools as a synonym for "scientific computing."

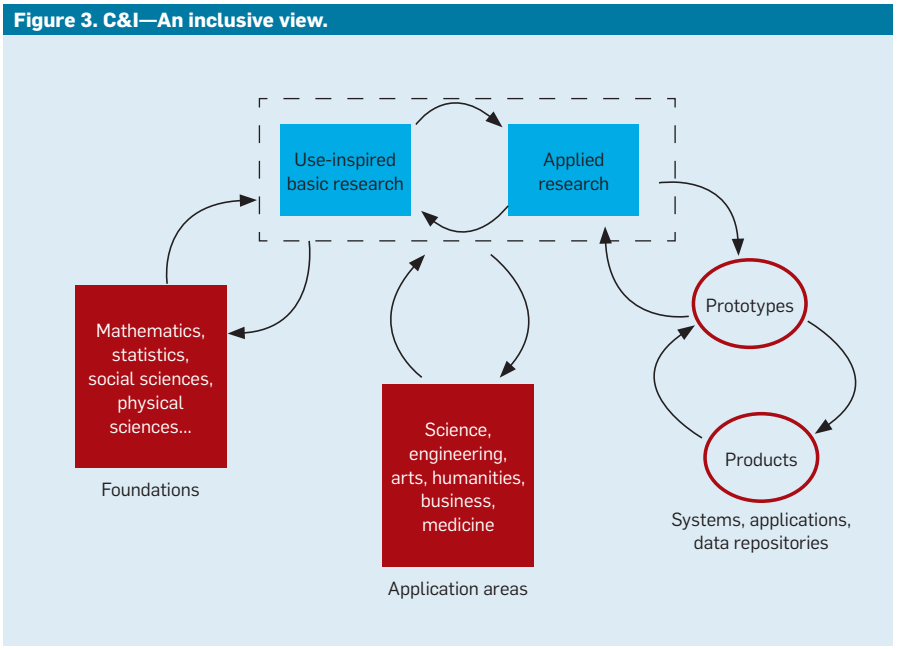
I believe, however, that *physical constraints are a small fraction of the constraints relevant to the design of C&I systems*. For example, software engineering research has strived for decades to define code metrics that represent how complex a code is (hence, what effort is required to program or debug it)—with limited success. Such a code metric would measure how difficult it is for a programmer to comprehend a code. But this is a cognitive issue: It is highly unlikely that one can develop successful theories on this subject without using empirically validated cognitive models that are based on our best understanding of human cognition. Unfortunately, traditional software engineering research has not been rooted in cognitive sciences.

Cognitive, cultural, social, organizational, and legal issues are increasingly important to engineering, in general.[5] This is a fortiori true for C&I. In the early days of computing, only few people interacted directly with computers: the psychology of programmers or users could be ignored without too much inconvenience: these few people would adapt to the computer. Today, the situation is vastly changed: Billions of people interact daily with digital devices and C&I systems become intimately involved in many cognitive and social processes; it is not possible anymore to ignore the human in the loop. Indeed, interesting research increasingly occurs at the intersection of the social and the technical: One may well argue that the essential insight that enabled efficient Web search and led to the creation of companies such as Google is that the structure of the Web carries information about the usefulness of Web pages—a socio-technical insight. Progress in graphics and animation increasingly requires an understanding of human vision: otherwise, one makes progress in quality metrics that have low correlation to the subjective quality of an image; examples can be easily multiplied.

Another important aspect of the evolution of our field is the increasing importance of applications. Precisely because software is so malleable and universal, one can develop very specialized systems to handle the needs of various disciplines: computer-aided design, medical imaging, DNA matching, Web auctions—these are but a few examples of application areas that have motivated significant specialized C&I research. Such research cannot be successful without a good understanding of the application area.

This suggests a new view for the organization of C&I that is described in Figure 3: Mathematics is no longer the only foundation. For those working close to hardware or working on cyber-physical systems, a good foundation in physics continues to be important. An increasing number of C&I research areas (such as human-computer interaction, social computing, graphics and visualization, and information retrieval) require insights from the social sciences (cognitive psychology, sociology, anthropology, economics, law, and so forth); human subject experiments become increasingly important for such research. At a more fundamental level, the development of artificial cognitive systems provides a better understanding of natural cognitive systems—of the brain and its function; and paradigms borrowed from C&I become foundational in biology. Insights from neuroscience provide a better way of building artificial intelligent systems and biology may become the source of future computing devices. Finally, research in C&I is strongly affected by the multiple application areas where information technology is used (such as science, humanities, art, and business), and profoundly affects these areas.

Figure 3. C&I—An inclusive view.



Organizational Implications

Similar to a school of medicine, a college of agriculture, or an engineering department, *I believe the correct organizational principle for a use-driven research area such as C&I is not common foundations, but shared concerns about the use of C&I systems*. The view illustrated in Figure 3 does not imply that each C&I researcher needs to be an expert in all core sciences or application areas. Rather, it implies that C&I researchers with different foundational knowledge and knowledge of different application domains will often need to work together in order to design, implement, and evaluate C&I systems and provide students with the education needed to do so.

The broad, integrated view of C&I is reflected at the NSF in the name of the Directorate for Computer and Information Science and Engineering (CISE). It is no surprise these days to find a linguist, anthropologist, or economist in a research lab at Microsoft or Yahoo. Some U.S. universities (including Carnegie-Mellon, Cornell, Georgia Tech, Indiana, Michigan, and the University of California at Irvine) are establishing or expanding schools or colleges that bring under one roof computer science, information science, applied informatics (C&I research that is application domain specific) as well as interdisciplinary research and education programs. These universities are still a minority. The broad, inclusive model is common in Japan (University of Tokyo, Kyoto University, Tokyo Institute of Technology, Osaka University), and is becoming more prevalent in the U.K. (Edinburgh, Manchester).

While organization models will differ from university to university, it is essential that all C&I units on a campus develop an integrative view of their field, and jointly develop coordinated research and education programs. This may require a change of attitude from all involved. Many cognitive and social aspects of system design are not amenable to quantitative studies; however, the engineering culture is often suspicious of social sciences and dismissive of qualitative sciences. Conversely, the importance of prototypes and artifacts is not always well appreciated outside engineering.

We can and should develop an environment where no scientist has an incentive to withhold information.

Undergraduate Curriculum

I discussed in the previous section the increasing variety of C&I research. In addition, there is a tremendous diversification of the professional careers in IT. Less than half of students who graduated in computer science in 1992–1993 were employed in traditional computer science professions 10 years after graduation (compared for 57% in engineering and 69% in health sciences).[4] In many computer science departments, more than half of the students graduating with bachelor's degrees are hired by companies in finance, services, or manufacturing, not by IT companies; this is where most of the growth in IT jobs is expected to be.[12] The Bureau of Labor Statistics tracks a dozen different occupations within computing[12] (although its categories are somewhat obsolete). A recent Gartner report[20] suggests the IT profession will split into four distinct professions: technology infrastructure and services, information design and management, process design and management, and relationship and sourcing management.

These trends imply an increasing diversification of C&I education. Currently, ABET accredits three different types of computing programs; ACM has developed recommendations for five curricula. Many schools experiment with more varied majors and interdisciplinary programs—in particular, Georgia Tech.[17] This evolution could lead to an increasing balkanization of our discipline: It is fair to assert that we are still more concerned with differentiating the various programs than defining their common content. In particular, should there be a core common to all programs in C&I?

To clarify: A common core is not about what every student in C&I must know: most of the specific knowledge we teach will be obsolete long before our students reach retirement age. A common core is about C&I "education,"[c] not about C&I know-how. It is about educating students in ways of thinking and problem solving that characterize our community and differentiate us from other communities: a system view of the world, a focus on mathematical and computational representations of systems, information representation and transformation, and so forth. The selection of courses for the core will not be based (only or mostly) on the usefulness of the facts taught, but on the skills and concepts that are acquired by the students.

I believe such a common core is extremely important: It is, to a large extent, what defines a discipline: You can expect a student of physics to take a sequence of physics courses that start with mechanics and end with quantum physics. This is not necessarily what those students will need in their future careers; but those courses define the physics canon. If we take ourselves seriously as a discipline, we should be able to define the C&I canon. Like physics, this core should be concise—say four courses: A common core does not preclude variety and specialization in junior and senior years.

Eating Our Own Dog Food

IT has a profound impact on the way the information economy works. It can and should have a profound impact on the operation of universities that are information enterprises par excellence. The C&I academic community can and should have a major role in pioneering this change. We should be ahead of the curve in using advanced IT in our professional life, and using it in ways that can revolutionize our enterprise. I illustrate the possibilities with a few examples here.

William J. Baumol famously observed that labor productivity of musicians has not increased for centuries: it still takes four musicians to play a string quartet.[2] This has become

c "Education is what remains after one has forgotten everything he learned in school"—A. Einstein.

known as "Baumol's cost disease:" Some sectors are labor intensive, require highly qualified personnel, and see no increases in labor productivity, due to improved technology. This is true for higher education: As long as a main measure of the quality of higher education is the student/faculty ratio, teaching productivity of faculty cannot increase; as long as faculty salaries keep up with inflation, the cost of higher education will keep up with inflation.[d] Such a situation will lead to the same pressures we see now in the health sector, and will force major changes. IT is, in many service sectors, the cure for Baumol's cost disease;[27] can it be in higher education?

IT often cures Baumol's cost disease not by increasing labor productivity, but by enabling a cheaper, replacement service. It still takes four musicians to play a string quartet, but digital recording enables us to enjoy the music where and when we want to hear it. ATMs replace bank tellers, Internet shopping replaces sales clerks. The convenience of getting a service where and when we want it, and the lower cost of self-service, compensate for the loss of personal touch. To many of our students, the idea that one must attend a lecture at a particular place and time in order to obtain a piece of information chosen by the lecturer is as antiquated as pre-Web shopping. Increasingly, students will want to obtain the information they need when and where they want it. An increasing shift to "self-service" education that is "student pull" based, rather than "lecturer push" based, may well be the cure to Baumol's cost disease in higher education, as well as the cure to the depressing passivity of many students.

"Self-service" education need not imply a lack of social interaction. The study of Richard J. Light, at Harvard, indicated that participation in a small student study group is a stronger determinant of success in a course than

d Note, however, that the recent fast rise in the cost of higher education in the U.S. is not due to increases in faculty salaries. According to the AAUP, faculty salaries have risen in real terms by 7% in the last three decades (http://www.aaup.org/AAUP/GR/CapHill/2008/risingcosts.htm); state support to public universities has shrunk by more than one-third during this period.[13]

the teaching style of the instructor.[20] Rather than focusing on the use of IT to improve the lecture experience, we should probably focus on the use of IT and social networking tools to make individual and group self-study more productive by multiplying the interaction channels between students and between students and faculty.[15]

As the half-life of knowledge grows shorter, it becomes less important to impart specific knowledge to students (and to test them on this knowledge) and more important to teach them how to learn, how to identify and leverage sources of knowledge and expertise, and how to collaborate with experts in other areas, creating collective knowledge. Yet our education is still strongly focused on acquiring domain-specific individual knowledge; and students mostly collaborate with other students that have similar expertise. Projects and practicums that involve teams of students from different programs, with different backgrounds, could refocus education so as to train more foxes and fewer hedgehogs[e]—a change I believe will benefit many of our students. Such collaborative learning-by-doing empowers students, increases motivation, improves retention and teaches skills that are essential for success in the information society. A skillful use of IT technology, both for supporting course activities and for assessing teaching and learning, can facilitate this education style.[f]

IT changes the way research is pursued: For example, it enables citizen science projects where many volunteers collect data. Such projects have become prevalent in environmental sciences[24] and are likely to have a large impact on health sciences. They not only provide researchers with data that cannot be obtained otherwise but also change in fundamental ways the relation of the scientist to the object of study. The volunteers are unlikely to be motivated by pure scientific curiosity; they want the research they participate in to have an impact—save the environment or cure cancer. The researcher that uses their data has an implicit or explicit obligation to use the data collected for that common purpose and not use it for other purposes. Research becomes engaged and obligated to a large community.[17]

IT enables the fast dissemination of scientific observations and results. Research progresses faster if observational data and preliminary results are shared as quickly and as broadly as possible. One obstacle to such unimpeded sharing is that academic careers are fostered by the publication of polished analyses, not by the publication of raw data or partial results: Research groups tend to hold on to their data until they can analyze it and obtain conclusive results. Better ways of tracking the provenance of data used by researchers and the web of mutual influences among researchers would enable to track the impact of contributions other than polished publications and develop a merit system that encourage more information sharing. We can and should develop an environment where no scientist has an incentive to withhold information.

C&I has been, for years, an amazingly vibrant, continuously renewing intellectual pursuit that has had a profound impact on our society. It has succeeded being so by continuously pursuing new uses of IT and continuously adjusting disciplinary focus in research and education so as to address the new problems. This fast evolution must continue for our discipline to stay vital. IT will continue to be a powerful agent of change in our society and, to drive this change, we must continuously change and strive to change our academic environment. C

e "The fox knows many things, but the hedgehog knows one big thing."[3] Sir Isaiah Berlin distinguishes between hedgehogs—thinkers "who relate everything to a single central vision," and foxes—thinkers who "pursue many ends, often unrelated and even contradictory, connected only in some de facto way." Although the essay of Isaiah Berlin focuses on Russian writers, I see "foxiness" as being very much the tradition of American Pragmatism. Both types are needed in our society, but "hedgehogs" who prize the hedgehog way of thinking seem to dominate in academia, especially in science and engineering.

f The recently started *International Journal on Computer-Supported Collaborative Learning* provides several useful references.

References

1. Atkinson, R.D. and McKay, A.S. *Digital Prosperity: Understanding the Economic Benefits of the Information Technology Revolution.* Information Technology and Innovation Foundation, 2007.
2. Baumol, W.J. and Bowen, W.G. *Performing Arts: The Economic Dilemma*, 1966.
3. Berlin, I. *The Hedgehog and the Fox.* Simon & Schuster, New York, 1953.
4. Choy, S.P., Bradburn, E.M., and Carroll, C.D. *Ten Years After College: Comparing the Employment Experiences of 1992–93 Bachelor's Degree Recipients With Academic and Career-Oriented Majors,* National Center for Education Statistics, Institute of Education Sciences, U.S. Department of Education, 2008.
5. Clough, G.W. *The Engineer of 2020: Visions of Engineering in the New Century.* National Academy of Engineering Press, Washington, D.C. (2004).
6. Cohen, W.M. and Merrill, S.A. *Patents in the Knowledge-based Economy.* National Academies Press, 2003.
7. Corrado, C.A., Sichel, D.E., and Hulten, C.R. *Intangible Capital and Economic Growth.* Board of Governors of the Federal Reserve System City, 2006.
8. Denning, P.J. Computer science: The discipline. *Encyclopedia of Computer Science* (2000).
9. Denning, P.J. Is computer science science? *Commun. ACM 48*, 4 (Apr. 2005), 27–31.
10. Denning, P.J. et al. Computing as a discipline. *Commun. ACM 32*, 1 (Jan. 1989).
11. Denning, P.J. and Freeman, P.A. Computing's paradigm. *Commun. ACM 52*, 12 (Dec. 2009), 28–30.
12. Dohm, A. and Shniper, L. Occupational employment projections to 2016. *Monthly Labor Review Online 130, 11* (Nov. 2007).
13. Ehrenberg, R. et al. Financial forces and the future of American higher education. *Academe 90*, 4 (Apr. 2004), 28–31.
14. Furst, M. and DeMillo, R.A. Creating symphonic-thinking computer science graduates for an increasingly competitive global environment. White Paper, College of Computing, Georgia Institute of Technology (2004).
15. Haythornthwaite C. et al. New theories and models of and for online learning. *First Monday 12*, 8 (August 2007).
16. Jorgenson, D.W., Ho, M.S. and Stiroh, K.J. *Information Technology and the American Growth Resurgence.* MIT Press, Cambridge, Mass., 2005.
17. Krasny, M.E. and Bonney, R. Environmental education through citizen science and participatory action research. *Environmental Education and Advocacy: Changing Perspectives of Ecology and Education* (2005), 292–319.
18. Light, R.J. *Making the Most of the College: Students Speak their Minds.* Harvard University Press, 2004.
19. Marrano, M.G., Haskel, J.E. and Wallis, G. *What Happened to the Knowledge Economy? ICT, Intangible Investment and Britain's Productivity Record Revisited.* Department of Economics, Queen Mary, University of London, 2007.
20. Morello, D. *The IT Professional Outlook: Where Will We Go From Here?* Gartner, 2005.
21. National Science Foundation—Directorate Computing and Information Science and Engineering. *CISE Pathways to Revitalized Undergraduate Computing Education (CPATH).* NSF, Arlington, VA.
22. National Science Foundation—Directorate Computing and Information Science and Engineering. *Cyber-Enabled Discovery and Innovation (CDI).* NSF, Arlington, VA.
23. Sell, S.K. *Private Power, Public Law: The Globalization of Intellectual Property Rights.* Cambridge University Press, 2003.
24. Silvertown, J. A new dawn for citizen science. *Trends in Ecology and Evolution 24*, 9 (Sept. 2009).
25. Stokes, D.E. *Pasteur's Quadrant.* Brookings Institution Press, 1997.
26. Thelwall, M. Can Google's PageRank be used to find the most important academic Web pages? *Journal of Documentation 59*, 2 (Feb. 2003), 205–217.
27. Triplett, J.E. and Bosworth, B.P. *'Baumol's disease' has been cured: IT and multifactor productivity in U.S. services industries.* Edward Elgar Publishing, City, 2006.
28. Wulf, W.A. The Urgency of Engineering Education Reform. *The Bridge 28*, 1 (Jan. 1998), 48.

Marc Snir (snir@illinois.edu) is Michael Faiman and Saburo Muroga Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign.

I thank Martha Pollack for her careful reading of an early version of this Viewpoint and for her many suggestions. Some of the ideas presented here were inspired by a talk by John King. This Viewpoint greatly benefited from the detailed feedback of one of the referees.

DOI:10.1145/1897852.1897868

Article development led by acmqueue
queue.acm.org

Models of determinism are changing IT management.

BY MARK BURGESS

Testable System Administration

THE METHODS OF system administration have changed little in the past 20 years. While core IT technologies have improved in a multitude of ways, for many if not most organizations system administration is still based on production-line build logistics (aka provisioning) and reactive incident handling—an industrial-age method using brute-force mechanization to amplify a manual process. As we progress into an information age, humans will need to work less like the machines they use and embrace knowledge-based approaches. That means exploiting simple (hands-free) automation that leaves us unencumbered to discover patterns and make decisions. This goal is reachable if IT itself opens up to a core challenge of automation that is long overdue—namely, how to abandon the myth of determinism and expect the unexpected.

We don't have to scratch the surface very hard to find cracks in the belief system of deterministic management. Experienced system practitioners know deep down that they cannot think of system administration as a simple process of reversible transactions to be administered by hand; yet it is easy to see how the belief stems from classical teachings. At least half of computer science stems from the culture of discrete modeling, which deals with absolutes as in database theory, where idealized conditions can still be simulated to an excellent approximation. By contrast, the stochastic models that originate from physics and engineering, such as queueing and error correction, are often considered too difficult for most basic CS courses. The result is that system designers and maintainers are ill prepared for the reality of the Unexpected Event. To put it quaintly, "systems" are raised in laboratory captivity under ideal conditions, and released into a wild of diverse and challenging circumstances. Today, system administration still assumes, for the most part, that the world is simple and deterministic, but that could not be further from the truth.

In the mid-1990s, several research practitioners, myself included, argued for a different model of system administration, embracing automation for consistency of implementation and using policy to describe an ideal state. The central pillar of this approach was stability.[2,4] We proposed that by placing stability center stage, one would achieve better reliability (or at the very least predictability). A tool such as IT is, after all, useful only if it leads to consistently predictable outcomes. This is an evolutionary approach to management: only that which survives can be successful.

As a physicist by training, I was surprised by the lack of a viable model for explaining actual computer behavior. It seemed that, instead of treating behavior as an empirical phenomenon full of inherent uncertainties, there was an implicit expectation that com-

ILLUSTRATION BY STUART BRADFORD

puters would behave as programmed. Everyone knows this to be simplistic; yet still, a system administrator would worry about behavior only when an incident reported something to the contrary.

From Demolition to Maintenance

When a problem occurs, many organizations take affected systems out of service, wipe them, and restore them from backup or reinstall from scratch. This is the only way they know to assure the state of the system because they know no simple way of discovering what changed without an arduous manual investigation. The process is crude, like tearing down a building to change a lightbulb. But the reason is understandable. Current tools are geared for building, not repairing—and if you've only got a sledgehammer...then you rebuild.

There is growing acceptance of a test-driven or diagnostic approach to the problem. This was originally ushered in by Cfengine,[5] and then partially adopted in other software such as Puppet.[11] In a test-driven approach, system state is regulated by continual reappraisal at a microscopic level, like having a groundskeeper watch continuously over an estate, plucking the weeds or applying a lick of paint where needed. Such an approach required the conceptual leap to a computable notion of *maintenance*. Maintenance can be defined by referring to a *policy* or *model* for an ideal system state. If such a model could somehow be described in terms of predictable, actionable repairs, in spite of environmental indeterminism, then automating maintenance would become a simple reality. This, in essence, is how Cfengine changed the landscape of IT management.

The term *compliance* is often used today for correctness of state with respect to a model. If a system deviates from its model, then with proper automation it *self*-repairs,[2,4] somewhat like an autopilot that brings systems back on course. What is interesting is that, when you can repair system state (both static configuration and runtime state), then the initial condition of the system becomes unimportant, and you may focus entirely on the desired outcome. This is the way businesses want to think about IT—in

terms of goals rather than "building projects"—thus also bringing us closer to a modern IT industry.

Convergence to a Desired State

Setting up a reference model for repair sounds like a simple matter, but it requires a language with the right properties. Common languages used in software engineering are not well suited for the task, as they describe sequential steps from a fixed beginning rather than end goals. Generally, we don't know the starting state of a machine when it fails. Moreover, a lot of redundant computation is required to track a model, and that would intrude on clarity. The way around this has been to construct declarative DSLs (domain-specific languages) that hide the details and offer predictable semantics. Although Cfengine was the first attempt to handle indeterminism, special languages had been proposed even earlier.[9]

Many software engineers are not convinced by the declarative DSL argument: they want to use the familiar tools and methods of traditional programming. For a mathematician or even a carpet fitter, however, this makes perfect sense. If you are trying to fit a solution to a known edge state, it is cumbersome to start at the opposite end with a list of directions that assume the world is fixed. When you program a GPS, for example, you enter the desired destination, not the start of the journey, because you often need to recompute the path when the unexpected occurs, such as a closed road. This GPS approach was taken by Cfengine[5] in the mid-1990s. It says: work relative to the desired end-state of your model, not an initial baseline configuration, because the smallest unexpected change breaks a recipe based on an initial state. This has been likened to Prolog.[7]

In simple terms, the approach works by making every change satisfy a simple algorithm:

$$\text{Change}(\text{arbitrary\_state}) \rightarrow \text{desired\_state} \quad (1)$$
$$\text{Change}(\text{desired\_state}) \rightarrow \text{desired\_state} \quad (2)$$

This construction is an expression of "dumb" stability, because if you perturb the desired state into some arbitrary state, it just gets pushed back into the desired state again, like an automated course correction. It represents a system that will recover from accidental or incidental error, just by repeating a dumb mantra—without the need for intelligent reasoning.

For example: suppose you want to reconfigure a Web server to support PHP and close a security hole. The server and all of its files are typically part of a software package and is configured by a complex file with many settings:

```
# >10kB of complex stuff
MODULES = SECURITY_HOLE JAVA
   OTHERS
# >10kb of complex stuff
```

To fix both problems, it is sufficient to alter only this list (for example, a desired outcome):

```
# >10kB of complex stuff
MODULES = JAVA OTHERS PHP
# >10kB of complex stuff
```

Traditionally, one replaces the whole file with a hand-managed template or even reinstalls a new package, forcing the end user to handle everything from the ground up. Using a desired state approach, we can simple say: in the context of file `webserver.config`, make sure that any line matching "`MODULES = something`" is such that "`something`" contains "`PHP`" and does not contain "`SECURITY HOLE`." Figure 1 illustrates how this might look in Cfengine.

Thus, the code defines two internal list variables for convenience and passes these to the specially defined method `edit_listvar`, which is constructed from convergent primitives. For each item in the list, Cfengine will assure the presence or absence of the listed atoms without touching anything else. With this approach, you don't need to reconstruct the whole Web server or know anything about how it is otherwise configured (for example, what is in "`complex stuff`") or even who is managing it: a desired end-state relative to an unknown start-state has been specified. It is a highly compressed form of information.

I referred to this approach as *convergent maintenance* (also likening the behavior to a human immune system[2]), as all changes converge on a destination or healthy state for the system in the frame of reference of the policy. Later, several authors adopted the mathematical term idempotence (meaning invariance under repetition), focusing on the fact that you can apply these rules any number of times and the system will only get better.

Guarded Policy

In the most simplistic terms, this approach amounts to something like Dijkstra's scheme of guarded commands.[8] Indeed, Cfengine's language implementation has as much in common with Guarded Command Lan-

Figure 1. Reconfiguring a Web server in Cfengine.

```
bundle agent webserver_config
{
vars:
 "add" slist => { "PHP", "php5" };
 "del" slist => { "SECURITY_HOLE", "otherstuff" };

column_edits:

  "APACHE_MODULES=.*"
     edit_column => edit_listvar("$(add_modules)","append");
  "APACHE_MODULES=.*"
     edit_column => edit_listvar("$(del_modules)","delete");
}

[Note: The syntax (which incorporates implicit guards and iteration)
has the form:

 type_of_promise:

   "Atom"

       property_type => desired_end_state;
]
```

guage as it does with Prolog.[7] The assertion of X as a statement may be interpreted as:

```
If not model(X), set model(X)
```

For example:

```
"/etc/passwd" create => "true";
```

Repeating an infinite number of times does not change the outcome. With hindsight, this seems like a trivial observation, and hardly a revolutionary technology, yet it is the simplest of insights that are often the hardest won. The implication of this statement is that X is not just what you want, but a model for what should be. The separation of the intended from the actual is the essence of the relativity.

There is one more piece to the puzzle: Knowing the desired state is not enough; we also have to know that it is achievable. We must add *reachability* of the desired state to the semantics.

Getting Stuck in Local Minima

It is well known from artificial intelligence and its modern applications that algorithms can get stuck while searching parameter landscapes for the optimum state. When you believe yourself at the bottom of the valley, how do you know there is not a lower valley just over the rise? To avoid the presence of false or local minima, you have to ensure that each independent dimension of the search space has only a single minimum, free of obstacles. Then there are two things at work: independence and convergence. Independence can be described by many names: atomicity, autonomy, orthogonality, and so on. The essence of them all is that the fundamental objects in a system should have no dependencies.

What we are talking about is a theory of policy atoms in an attribute space. If you choose vectors carefully (such as, file permissions, file contents, and processes) so that each change can be made without affecting another, no ordering of operations is required to reach a desired end-state, and there can be only one minimum. Indeed order-independence can be proven with periodic maintenance as long as the operators form irreducible groups.[3,6]

The discovery of such a simple solution suggests a panacea, ushering in a new and perfect world. Alas, the approach can be applied only partially to actual systems because no actual systems are built using these pure constructions. Usually, multiple change mechanisms tether such atoms together in unforeseeable ways (for example, packages that bundle up software and prevent access to details). The approximation has worked remarkably well in many cases, however, as evidenced by the millions of computers running this software today in the most exacting environments. Why? The answer is most likely because a language that embodies such principles encourages administrators to think in these terms and keep to sound practices.

If you are trying to fit a solution to a known edge state, it is cumbersome to start at the opposite end with a list of directions that assume the world is fixed.

Tangled by Dependency: The Downside of Packaging

The counterpoint to this free atomization of system parts is what software designers are increasingly doing today: bundling atoms and changes together into packages. In modern systems packaging is a response to the complexity of the software management process. By packaging data to solve one management problem, however, we lose the resolution needed to customize what goes on inside the packages and replace it with another. Where a high degree of customization is needed, unpacking a standard "package update" is like exploding a smart bomb in a managed environment—wiping out customization—and going back to the demolition school of management.

We don't know whether any operating system can be fully managed with convergent operations alone, nor whether it would even be a desirable goal. Any such system must be able to address the need of surgically precise customization to adapt to the environment. The truly massive data centers of today (Google and Facebook) are quite monolithic and often less complex than the most challenging environments. Institutions such as banks or the military are more representative, with growth and acquisition cultures driving diverse challenges to scale. What *is* known is that no present-day operating system makes this a completely workable proposition. At best one can approximate a subset of management operations, but even

this leads to huge improvements in scalability and consistency of process—by allowing humans to be taken out of the process.

From Demanding Compliance To Offering Capability

What is the future of this test-driven approach to management? To understand the challenges, you need to be aware of a second culture that pervades computer science: the assumption of management by *obligation*. Obligations are modal statements: for example, X must comply with Y, A should do B, C is allowed to do D, and so on. The assumption is that you can force a system to bow down to a decision made externally. This viewpoint has been the backbone of policy-based systems for years,[12] and it suffers from a number of fundamental flaws.

The first flaw is that one cannot generally exert a mandatory influence on another part of a software or hardware system without its willing consent. Lack of authority, lack of proximity, lack of knowledge, and straightforward impossibility are all reasons why this is impractical. For example, if a computer is switched off, you cannot force it to install a new version of software. Thus, a model of maintenance based on obligation is, at best, optimistic and, at worst, futile. The second point is that obligations lead to contradictions in networks that cannot be resolved. Two different parties can insist that a third will obey quite different rules, without even being aware of one another.[1]

Realizing these weaknesses has led to a rethink of obligations, turning them around completely into an atomic theory of "voluntary cooperation," or promise theory.[1] After all, if an obligation requires a willing consent to implement it, then voluntary cooperation is the more fundamental point of view. It turns out that a model of promises provides exactly the kind of umbrella under which all of the aspects of system administration can be modeled. The result is an agent-based approach: each system part should keep its own promises as far as possible without external help, expecting as little as possible of its unpredictable environment.

Independence of parts is represented by agents that keep their own promises; the convergence to a standard is represented by the extent to which a promise is kept; and the insensitivity to initial conditions is taken care of by the fact that promises describe outcomes, not initial states.

Promise theory is a wide-ranging description of cooperative model building that thinks bottom-up instead of top-down. It can be applied to humans and machines in equal measure and can also describe human workflows—a simple recipe for federated management.

Promise theory turns out to be a rather wide-ranging description of cooperative model building that thinks bottom-up instead of top-down. It can be applied to humans and machines in equal measure and can also describe human workflows—a simple recipe for federated management. It has not yet gained widespread acceptance, but its principal findings are now being used to restructure some of the largest organizations in banking and manufacturing, allowing them to model complexity in terms of robust intended states. Today, only Cfengine is intentionally based on promise theory principles, but some aspects of Chef's decentralization[10] are compatible with it.

The Limits of Knowledge

There are subtler issues lurking in system measurement that we've only glossed over so far. These will likely challenge both researchers and practitioners in the years ahead. To verify a model, you need to measure a system and check its compliance with the model. Your assessment of the state of the system (does it keep its promises?) requires a trust of the measurement process itself to form a conclusion. That one dependence is inescapable.

What happens when you test a system's compliance with a model? It turns out that every intermediate part in a chain of measurement potentially distorts the information you want to observe, leading to less and less certainty. Uncertainty lies at the very heart of observability. If you want to govern systems by pretending to know them absolutely, you will be disappointed.

Consider this: environmental influences on systems and measurers can lead directly to illogical behavior, such as undecidable propositions. Suppose you have an assertion (for example, promise that a system property is true). In logic this assertion must either be true or false, but consider these cases:

- You do not observe the system (so you don't know);
- Observation of the system requires interacting with it, which changes its state;

▸ You do not trust the measuring device completely; or

▸ There is a dependence on something that prevents the measurement from being made.

If you believe in classic first-order logic, any assertion must be either true or false, but in an indeterminate world following any of these cases, you simply do not know, because there is insufficient information from which to choose either true or false. The system has only two states, but you cannot know which of them is the case. Moreover, suppose you measure at some time *t*; how much time must elapse before you can no longer be certain of the state?

This situation has been seen before in, of all places, quantum mechanics. Like Schrodinger's cat, you cannot know which of the two possibilities (dead or alive) is the case without an active measurement. All you can know is the outcome of each measurement reported by a probe, after the fact. The lesson of physics, on the other hand, is that one can actually make excellent progress without complete knowledge of a system—by using guiding principles that do not depend on the uncertain details.

Back to Stability?

A system might not be fully knowable, but it can still be self-consistent. An obvious example that occurs repeatedly in nature and engineering is that of *equilibrium*. Regardless of whether you know the details underlying a complex system, you can know its stable states because they persist. A persistent state is an appropriate policy for tools such as computers—if tools are changing too fast, they become useless. It is better to have a solid tool that is almost what you would like, rather than the exact thing you want that falls apart after a single use (what you want and what you need are not necessarily the same thing). Similarly, if system administrators cannot have what they want, they can at least choose from the best we can do.

Systems can be stable, either because they are unchanging or because many lesser changes balance out over time (maintenance). There are countless examples of very practical tools that are based on this idea: Lagrange points (optimization), Nash equilibrium (game theory), the Perron-Frobenius theorem (graph theory), and the list goes on. If this sounds like mere academic nonsense, then consider how much of this nonsense is in our daily lives through technologies such as Google PageRank or the Web of Trust that rely on this same idea.

Note, however, that the robustness advocated in this article, using the principle of atomization and independence of parts, is in flat contradiction with modern programming lore. We are actively encouraged to make hierarchies of dependent, specialized objects for reusability. In doing so we are bound to build fragilities and limitations implicitly into them. There was a time when hierarchical organization was accepted wisdom, but today it is becoming clear that hierarchies are fragile and unmanageable structures, with many points of failure. The alternative of sets of atoms promising to stabilize patches of the configuration space is tantamount to heresy. Nevertheless, sets are a more fundamental construction than graphs.

For many system administrators, these intellectual ruminations are no more pertinent than the moon landings were to the users of Teflon pans. They do not see themselves in these issues, which is why researchers, not merely developers, need to investigate them. Ultimately, I believe there is still great progress to be made in system administration using these approaches. The future of system administration lies more in a better understanding of what we already have to work with than in trying to oversimplify necessary complexity with industrial force.

Conclusion

It is curious that embracing uncertainty should allow you to understand something more fully, but the simple truth is that working around what you don't know is both an effective and low-cost strategy for deciding what you actually can do.

Major challenges of scale and complexity haunt the industry today. We now know that scalability is about not only increasing throughput but also being able to comprehend the system as it grows. Without a model, the risk of not knowing the course you are following can easily grow out of control. Ultimately, managing the sum knowledge about a system is the fundamental challenge: the test-driven approach is about better knowledge management—knowing what you can and cannot know.

Whether system administration is management or engineering is an oft-discussed topic. Certainly without some form of engineering, management becomes a haphazard affair. We still raise computers in captivity and then release them into the wild, but there is now hope for survival. Desired states, the continual application of "dumb" rule-based maintenance, and testing relative to a model are the keys to quantifiable knowledge. C

Related articles on queue.acm.org

A Plea to Software Vendors from Sysadmins—10 Do's and Don'ts
Thomas A. Limoncelli
http://queue.acm.org/detail.cfm?id=1921361

Self-Healing in Modern Operating Systems
Michael W. Shapiro
http://queue.acm.org/detail.cfm?id=1039537

A Conversation with Peter Tippett and Steven Hofmeyr
January 10, 2009
http://queue.acm.org/detail.cfm?id=1071725

References

1. Burgess, M. An approach to understanding policy based on autonomy and voluntary cooperation. Submitted to *IFIP/IEEE 16th International Workshop on Distributed Systems Operations and Management* (2005).
2. Burgess, M. Computer immunology. In *Proceedings of the 12th System Administration Conference*, 1998.
3. Burgess, M. Configurable immunity for evolving human-computer systems. *Science of Computer Programming 51*, 3 (2004), 197–213.
4. Burgess, M. On the theory of system administration. *Science of Computer Programming 49* (2003), 1–46.
5. Cfengine; http://www.cfengine.org.
6. Couch, A., Daniels, N. The maelstrom: network service debugging via `ineffective procedures.' *Proceedings of the 15th Systems Administration Conference* (2001), 63.
7. Couch, A., Gilfix, M. It's elementary, dear Watson: Applying logic programming to convergent system management processes. In *Proceedings of the 13th Systems Administration Conference* (1999), 123.
8. Dijkstra, E. http://en.wikipedia.org/wiki/Guarded_Command_Language.
9. Hagemark, B., Zadeck, K. Site: A language and system for configuring many computers as one computer site. *Proceedings of the Workshop on Large Installation Systems Administration III* (1989); http://www2.parc.com/csl/members/jthornton/Thesis.pdf.
10. Opscode; http://www.opscode.com/chef.
11. Puppet Labs; http://www.puppetlabs.com/.
12. Sloman, M. S., Moffet, J. Policy hierarchies for distributed systems management. *Journal of Network and System Management 11*, 9 (1993), 404.

Mark Burgess is a professor of network and system administration, the first with this title, at Oslo University College. His current research interests include the behavior of computers as dynamic systems and applying ideas from physics to describe computer behavior. He is the author of Cfengine and is the founder, chairman, and CTO of Cfengine, Oslo, Norway.

DOI:10.1145/1897852.1897869

Article development led by acmqueue
queue.acm.org

Attacks in Estonia and Georgia highlight key vulnerabilities in national Internet infrastructure.

BY ROSS STAPLETON-GRAY AND WILLIAM WOODCOCK

National Internet Defense—Small States on the Skirmish Line

DESPITE THE GLOBAL and borderless nature of the Internet's underlying protocols and driving philosophy, there are significant ways in which it remains substantively territorial. Nations have policies and laws that govern and attempt to defend "their Internet"—the portions of the global network that they deem to most directly impact their commerce, their citizens' communication, and their national means to project social, political, and commercial activity and influence. This is far less palpable than a nation's physical territory or even than "its air" or "its water"—one could, for example, establish by treaty how much pollution Mexican and American factories might contribute to the atmosphere along their shared border, and establish metrics and targets fairly objectively. Cyberspace is still a much wilder frontier, difficult to define and measure. Where its effects are noted and measurable, all too often they are hard to attribute to responsible parties.

Nonetheless, nation-states are taking steps to defend that space, and some have allegedly taken steps to at-

ILLUSTRATION BY ALEX WILLIAMSON

tack that of others. Two recent events illustrate the potential vulnerabilities faced by small nation-states and suggest steps that others may take to mitigate those vulnerabilities and establish a more robust and defensible Internet presence. The first was an attack on Estonian Internet infrastructure and Web sites in May and June 2007. The second was a cyber attack against the Georgian infrastructure that accompanied the Russian incursion into South Ossetia in August 2008.

Estonia

Tensions had been building in Estonia in the spring of 2007 over the country's plans to relocate the Bronze Soldier, a Soviet war memorial, and the capital, Tallinn, experienced several nights of rioting. The subsequent cyber attacks are believed to be a consequence of the memorial's relocation.

An attack on Estonian Internet infrastructure and Web sites began at 11 P.M. local time, midnight Moscow time, Tuesday, May 8. The attack was effectively mitigated by 7 A.M. the following day but continued to be visible in traffic logs for exactly 30 days thereafter. That time period, together with the fact that the attacking botnets' signature was identical to that used in prior Russian Business Network spam-sending campaigns, suggests that this was a one-month attack for hire (or was intended to look like one). Unfortunately, such attacks, either threatened or launched for commercial extortion, have become commonplace. Based on offers visible on the black market at the time, the attack likely cost between $200 and $2,000 to hire. Like many politically motivated attacks, it combined a distributed denial-of-service (DDoS) attack against Internet infrastructure with DDoS and attempted defacement attacks against the Web sites of Estonian banks, media outlets, and government.

The Estonian defense was notably

successful, and there are a number of lessons to be taken from it by other countries wishing to avoid a cyberwarfare defeat. The simplest summary of the dynamics of a DDoS-based cyber attack is as a numbers game. An attacker with greater network capacity than the defender will be able to overwhelm the defender's network, while retaining sufficient capacity to support its own needs at the same time. Such an attack would be deemed successful. An attacker with less bandwidth than the defender would exhaust itself in consuming the defender's capacity, while the defender might well retain enough excess capacity that its population would not be significantly inconvenienced; such an attack would be considered unsuccessful.

Viewed in closer detail, there are different kinds of network capacity and different mechanisms for improving and defending each. They can be placed in four categories: local or internal capacity; external connectivity; name resolution capability; and defensive coordination.

Local capacity, or bandwidth, is most familiar as one's initial connection to the Internet. This local loop, or last mile, is the copper wire or fiber line in the ground or on poles, or the wireless link that carry signals from the customer to an ISP (Internet service provider). A robust local-loop infrastructure consists of buried fiber-optic cable interconnecting each business or residence with multiple ISPs over different physical paths. Ideally, these service providers ought to be in competition so they cannot be collectively suborned or sabotaged, and so their prices are low enough that people can actually choose fluidly among them. A sparsely supplied market for local connectivity can create bottlenecks and make attractive targets. In Estonia's case, multiple independent fiber infrastructure operators existed, and many different ISPs built a healthy, competitive marketplace on top of that. More—and more diverse—domestic fiber is always better, but Estonia's was more than sufficient.

External connectivity. More important to defensibility is the ecosystem for the providers' own connectivity within that domestic context. The modern means to create an effective mesh of providers is via Internet exchange points, commonly abbreviated IXP. The world has about 330 IXPs at the moment, and that number has been steadily increasing. Each IXP has a specific physical location and connects a community of ISPs that meet as peers at the exchange. Some countries, such as the U.S., have many IXPs. Others, such as the Netherlands and Germany, have very large IXPs. Many smaller countries have exactly one exchange, located in the capital city. But the greatest number of countries, typically the smallest ones, has no IXP at all. This means that they are heavily dependent for their *domestic* connectivity upon *international* data circuits. Imagine a situation in which there were no local telephone calls, only calls overseas; to reach someone next door, you would have to make a call that went overseas *and then back again*, at twice the cost.

This is the situation in most less-developed countries, as a result of misunderstanding Internet economics and topology. Countries in this situation are extremely vulnerable to having those external lines of communications cut or overburdened, since that causes not only international but also domestic communications to fail, and thus the ability to coordinate a defense fails as well. A strong domestic Internet exchange point is the first and most critical component of a cyberwarfare defense. A redundant pair of IXPs, or one in each major city, is the desirable goal. A redundant pair of IXPs in Tallinn formed the linchpin of the Estonian defense.

International communications capability is necessary for conducting business in a global economy. It's also needed for defensive coordination with outside allies in order to protect a nation's international capacity. International capacity is the asset most easily targeted from the outside, and it is perhaps the most challenging to defend from the perspective of the state, since it's a multinational private-sector resource. In most countries, each circuit that crosses the border is controlled by one company at one end, another company at the other end, and a third in between. In turn, many of these companies are themselves consortia of other multinational companies. On the domestic end of a circuit regulatory jurisdiction is generally clear, though limited and perhaps difficult to enforce; but on the other end it is nearly impossible even to influence. Thus, diversity is key to optimizing the survivability of international connectivity.

Estonia had numerous privately controlled data circuits crossing its borders, with the other ends located in several different countries. Of these, the most significant were large Scandinavian and Western European ISPs with which Estonian ISPs had commercial relationships and that were based in diplomatically friendly neighboring countries. This is an optimal situation, and when push came to shove, Estonia received fast and effective aid from the ISPs at the other ends of those circuits.

Name resolution. The ability to resolve domain names domestically is another critical infrastructure capability. The Domain Name System (DNS) is the Internet's directory service, providing Internet-connected computers with the ability to map the human-readable domain names in email and Web addresses to the machine-readable binary IP addresses used to route traffic within the network. Domain names are resolved to IP addresses (and vice versa) by iterating through a delegation hierarchy of DNS directory servers, starting at the "root" and progressing through top-level domain (TLD) name servers such as .com and .net, to the organization-specific name servers that hold the particular answer one is looking for.

If connectivity is broken between users and any one of the name servers in the delegation chain from the root down to the specific one they are looking for, then the users will be unable to resolve the domain name they're looking for, and unable to reach the corresponding Web site or send the email, regardless of whether they have connectivity to the Web site or email addressee. If the directory service is broken, you can't *find* things, even if you could, hypothetically, *reach* them. Estonia did not have any root servers within the country at the time of the attack, and still does not today. This is one of the few weak points of the Estonian defense and would have become more debilitating over the course of an attack that had been more effective for a longer period of time.

Defensive coordination. The final component of an effective cyberwarfare defense is coordination. Knowing that one is under attack is an intelligence function. Identifying and characterizing the attack is a forensic analytical function. Communicating this information to the ISPs that can mitigate the attack is a communications function. These functions are most often coordinated by a computer emergency response team (CERT), or sometimes called a CIRT (computer incident response team). A CERT is the glue that holds a defense together, providing expertise, analytical facilities, and open lines of communication between the many organizations that are party to the defense or have some stake in its success.

CERTs provide training and preparedness workshops, maintain and exercise contact lists, and observe trends and find patterns in online criminal, military, and espionage activity. When a country is under attack, CERTs help individual organizations identify which portions of the attack are directed against them particularly, as opposed to those that they're feeling the effects of incidentally. CERTs provide the expertise to help those organizations with the very specialized tasks of discerning attack traffic from legitimate traffic and developing filters that will block the attack while protecting their ability to conduct business. CERTs will then communicate those filters up the path of ISPs toward the attackers, blocking the malicious traffic at each step, pushing the boundary of the cleaned network away from the victims and toward the attackers.

Georgia

A little more than a year after the Estonian incident, Georgia was subjected to cyber attacks in conjunction with the Russian incursion into South Ossetia in August 2008. This more complex attack combined Georgian targets with domestic media outlets that were perceived to be reporting news from a Georgian perspective.

Much of what had worked well in the case of Estonia did not in the Georgia attack. Relative to Estonia, Georgia suffered from two crippling deficiencies: Georgian international connectivity was far more limited, hence more vulnerable. Most of its international links were through Russian territory; and unlike Estonia, Georgia had no IXPs. As with Estonia, Georgia lacked a DNS root server, but that was mooted by its limited infrastructure being easily overwhelmed.

A sparsely supplied market for local connectivity can create bottlenecks and make attractive targets.

Given the relatively modest infrastructure and comparative lack of e-commerce to be affected (and all dwarfed in significance by an actual shooting war), it may be more difficult to extract lessons from Georgia's experience than from Estonia's. One noteworthy issue in the case of Georgia, however, was the number of offers made by governments and corporations to "mirror" Georgian Web content. If the Georgian government desired to reach a non-Georgian audience for sympathy and support, then distributing that message to parties outside Georgia and in regions of the Internet far less amenable to denial-of-service attacks would be a worthwhile strategy.

Why Cyberwar?

The mere fact that significant conversation is still occurring more than three years after the attacks on Estonia indicates that even if the destructive impact was minimal, the overall information warfare effect was significant. The return on a very small investment was disproportionately high; these margins suggest that cyberwarfare techniques will continue to be applied until they become considerably more expensive or less noticed.

It is worth understanding what was successful about the attack and what was successful about the defense. Viewed in the large, the Chinese cyberwarfare doctrine upon which the attacks were patterned states that one of the principal goals of an attack is to dispirit an adversary's civilian population, reduce their productivity, and cause them to withdraw economic, and eventually moral, support from their country's engagement in the conflict. This was not the SCADA attack—an attack on the cyber aspects of physical systems, with the intent to cripple the latter—that is so often warned of in the U.S. (SCADA, for supervisory control and data acquisition, is a catchall label for the various systems used to manage industrial

systems and processes, from factories to pipelines to transportation networks.) Rather, The Estonia incident was a pure information-warfare attack, attempting to convince Estonians that the information-economy infrastructure of which they were so proud was vulnerable and unsound, that their work in that sector was of little value, that their adversary was more capable and better prepared, and in a more pitched conflict, their defeat would be inevitable. A population that would take such a message to heart would indeed be unwilling to support conflict against the attacker.

The Estonia attack had very little success in concrete terms, and little more success in information-warfare terms, relative to the Estonians against whom it was directed. Because of its apparent state-on-state nature, and Estonia's status at the time as the most recently admitted NATO ally, the attack managed to garner a surprising degree of attention elsewhere, though. The attacks against Georgia were far more effective, but Georgia did not have as far to fall and the conflict on the Internet paled in comparison to the actual shooting war in its territory. One might accurately term both the Estonia and Georgia cyber assaults as skirmishing; the attack on Estonia amounted to little more than a nuisance, in part because of its scale and in part because of the effectiveness of the response.

Without a doubt, any major war would see complementary attacks against the adversaries' information infrastructure, including their national presence on the Internet—suppression of the means to coordinate and organize has long been a basic tenet of warfare. It is perhaps early to assess the impact of cyberwar, absent "real war"; the attack against Estonia was too slight to measure significant effects, while the attack on Georgia was just a sideshow to a widely, physically destructive conflict.

The ultimate source of both attacks remains murky. Many assertions have been made, but there has been little actual discussion of the question of state involvement in cyber attacks. Plausible deniability has become the watchword in cyberwarfare, and accordingly, attribution has become a major focus of effort, consuming far more resources than does actual defense.

Much of what had worked well in the case of Estonia did not in the Georgia attack...Georgian international connectivity was far more limited, hence more vulnerable.

Defending the Small Nation-State

Ensuring the Internet security of a small nation-state entails investment in four areas: ensuring physical network robustness; securing the interconnection of participating networks through exchange points; securing the data and services required to keep the Internet running; and developing an effective response community.

In advance of any threat, a nation should take steps to ensure that its networks are connected to the rest of the world via diverse international transit links to different unrelated transit providers in different, unaligned countries. A significant factor in why Georgia was so affected by its cyber attack was its extremely limited connectivity to the outside world; Estonia was in a far better position, with a more diverse mesh of connectivity to friendlier neighbors. Submarine cables are also worth noting as a clear point of vulnerability in international transit. There have been a number of accidental submarine cable cuts in the past several years, and a coordinated, willful effort to take those out would be fairly simple to mount and would have significant effect in certain regions.

In the case of Estonia, DoS attacks effectively stopped at the country's IXP and had minimal impact on domestic Internet traffic. In countries lacking IXPs, even domestic traffic may end up routed internationally, at greater expense than if there had been an IXP to broker exchanges before incurring higher international transit costs, and at greater risk of disruption.

It is critical that countries have root and TLD name servers well connected to their domestic IXPs, such that all of their domestic ISPs can provide uninterrupted DNS service to their customers. In the case of ISO country-code TLD name servers, such as those for Estonia's .ee domain, that's relatively easily accomplished, though not yet universally done. In the case of root name servers, it requires the cooperation and goodwill of a foreign organization, the operator of the root name server, and generally some small investment in infrastructure support for the remotely operated root server. This

might amount to an expenditure of some $15,000 (U.S.) per year, per root server installation within the country.

(It's worth noting that all of the investments required for cyberwarfare defense are equally applicable to general economic development. Just as the cyberwarfare field of conflict is a private-sector space, this, too, is unlike traditional military expenditures. A tank or a bunker is purely a cost center, whereas an IXP or domain name server is a profit center, generating new, concrete, and monetized value for its users from the moment it's established. The return on investment of a newly established IXP is typically less than three weeks, and often less than one week.)

The CERT is a widely employed model for computer and network incident response. CERTs are directly responsible for systems under their own control, and, with other CERTs, collaborate on collective network security. FIRST (Forum of Incident Response and Security Teams), an association of CERTs, brings CERTs and their staffs together to build the most fundamental links in a web of trust.[1] A CERT should also have already established lines of communication with ISPs, law enforcement, and other elements of government concerned with infrastructure security.

Network operators' groups promote community and cooperation between a country's Internet operators and their foreign counterparts. Participation in Inter-network Operations Center Dial-by-ASN (INOC-DBA) and Network Service Provider Security (NSP-SEC) can also aid in coordinating incident response. INOC-DBA is a voice over Internet Protocol (VoIP) hotline system, interconnecting network operation centers; it uses the networks' own numeric identifiers as dialing numbers so that a NOC operator observing problematic traffic can merely enter the address of the offending network to place a call to the responsible party.[2] NSP-SEC is an informal organization of security professionals at the largest Internet infrastructure providers: "Membership in NSP-SEC is restricted to those actively involved in the mitigation of [Network Service Provider] security incidents within organizations in the IP transit, content, and service provider community. Therefore, it will be limited to operators, vendors, researchers, and people in the FIRST community working to stop NSP security incidents."[3]

New members of the "culture of security" come out of academic and training programs (which must be established), intern in a CERT (internationally or domestically), and go on to careers as CSOs (chief security officers) in CERTs, academia, law enforcement, or government. This is fundamentally analogous to the peopling of a national health environment with doctors.

In the U.S., the Department of Homeland Security has included CERTs and information assurance analysts and operators in a new research and development solicitation. In a draft of the solicitation, DHS notes, "While we have a good understanding of the technologies involved in [cybersecurity incident response teams], we have not adequately studied the characteristics of individuals, teams, and communities that distinguish the great [cybersecurity incidence] responders from the average technology contributor. In other areas where individual contributions are essential to success, for example, first responders, commercial pilots, and military personnel, we have studied the individual and group characteristics essential to success. To optimize the selection, training, and organization of CSIR personnel to support the essential cyber missions of DHS, a much greater understanding and appreciation of these characteristics must be achieved."

Conclusion

It would be fair to describe these two incidents—Estonia in 2007, and Georgia a year later—as "cyberskirmishing." The attacks on Estonia amounted to little more than a nuisance, though a quite visible and much discussed one. Georgia had far greater problems to deal with in an armed incursion into its territory, and the Internet was not a factor in that fight.

The difference in responsiveness between the two, however, recommends that the small nation-state ought to make investments in Internet defensibility akin to those seen in Estonia:

- Through policy and regulation, and perhaps government investment, foster a robust physical infrastructure.
- Similarly, take steps to ensure a diversity of international connections.
- Encourage (or directly sponsor) creation of one or more IXPs.
- Ensure the domestic availability of DNS resolution, through root servers.
- Foster the growth of a collaborating community of security professionals.

A diversity of interconnections, both international and domestic, facilitated by the efficient peering afforded by IXPs, provides a more robust logical infrastructure, and local DNS resolution further lessens dependence on more exposed international connections. With that technical infrastructure ensured, nations should then foster development of the human infrastructure, the information security personnel needed to anticipate threats, the ability to intercede inventively to restore services, and the ability to support incident forensic collection and analysis. C

Related articles on queue.acm.org

Cybercrime 2.0: When the Cloud Turns Dark
Niels Provos, Moheeb Abu Rajab, Panayiotis Mavrommatis
http://queue.acm.org/detail.cfm?id=1517412

CTO Roundtable: Malware Defense
http://queue.acm.org/detail.cfm?id=1731902

The Evolution of Security
Daniel E. Geer
http://queue.acm.org/detail.cfm?id=1242500

References

1. FIRST; http://first.org/about/.
2. Inter-network Operations Center Dial-by-ASN (INOC-DBA), a Resource for the Network Operator Community; http://www2.computer.org/portal/web/csdl/doi/10.1109/CATCH.2009.36.
3. NSP Security Forum; http://puck.nether.net/mailman/listinfo/nsp-security.

Bill Woodcock is a founder and research director of Packet Clearing House, a nonprofit research institute dedicated to understanding and supporting Internet traffic exchange technology, policy, and economics. He entered the field of Internet routing research in 1989 while serving as the network architect and operations director for an international multiprotocol service-provision backbone network. Woodcock has participated in the establishment of more than 70 public Internet exchange points in Europe, Africa, Asia, and the Americas.

Ross Stapleton-Gray is research program manager at Packet Clearing House. Prior to joining PCH, he served as an intelligence analyst for the CIA, in information policy positions with the American Petroleum Institute and the University of California Office of the President, and has worked with several IT security start-ups, including as a cofounder of Sandstorm Enterprises.

DOI:10.1145/1897852.1897870

Article development led by acmqueue
queue.acm.org

Why can't we all use standard libraries for commonly needed algorithms?

BY POUL-HENNING KAMP

B.Y.O.C (1,342 Times and Counting)

ALTHOUGH SELDOM ARTICULATED clearly, or even at all, one of the bedrock ideas of good software engineering is reuse of code libraries holding easily accessible implementations of common algorithms and facilities. The reason for this reticence is probably because there is no way to state it succinctly, without sounding like a cheap parody of Occam's razor: *Frustra fit per plura quod potest fieri per pauciora* (it is pointless to do with several where few will suffice).

Obviously, choice of programming language means that "few" will never be "a single one," and until somebody releases a competent implementation under an open source license, we may have several more versions floating around than are strictly necessary, for legal rather than technological reasons.

It also never hurts to have a few competing implementations to inspire improvement; in fact, there seems to be a distinct lack of improvement where a single implementation becomes too "golden."

So much for theory. Does any of this hold in practice?

One of the nice side effects of the "software tools" concept is that programs are data, too. We can apply data mining methods to program source code, allowing us to investigate such questions.

Cryptography algorithms provide a good example because they are easier to identify than other algorithms. Magic numbers in crypto algorithms make for good oracular answers to their presence: you are not likely to encounter both 0xc76c51a3 and 0xd192e819 anywhere other than an implementation of SHA-2. Creating an oracle to detect sorting algorithms in source code with (p>0.9) would be a good student project (albeit, likely impossible).

For data mining FOSS (free and open source software) programs, the FreeBSD operating system ships with a handy facility called the Ports Collection, containing strategic metadata for 22.003 pieces of FOSS. A small number of these "ports" are successive versions of the same software (Perl 5.8, Perl 5.10, among others), but the vast majority are independent pieces of software, ranging from trivialities such as XLogo to monsters such as Firefox and OpenOffice.

A simple command downloads and extracts the source code to as many ports as possible into an easily navigated directory tree:

```
cd /usr/ports ; make -k extract
```

You will obviously need both sufficient disk space and patience. (`Using cd /usr/ports ; make -k -j 10 extract` will do 10 pieces of software in parallel, but will be a bandwidth hog.)

The results are worse. I had not expected to see 1,342, as shown in the accompanying table.[a] I expect that these numbers will trisect my readers into three somewhat flippantly labeled seg-

a Sorry, I forgot to include the DES algorithm in the search.

Crypto algorithms search results.

Cryptographic Implementations	
Algorithm	**Detected**
MD2	6
MD4	49
MD5	920
SHA-1	136
SHA-2	192
AES	39
Total	**1,342**

ments: "huh?," "sigh," and "aghast."

The "huh?" segment wonders what the big deal is: the absence of a standardized system library with these functions means that you have to "Bring Your Own Crypto" if you want some.

The "sigh" segment thinks this is the least of our troubles.

The "aghast" segment will see this as a total failure of good software engineering practices, a call to arms for better education, and reason for a stake through the heart of the Open Zombie Group.

And they are all correct, of course, each from its own vantage point.

Fortunately, what this is not, is The Next Big Security Issue, even though I would not be surprised if one or more "security researchers" would claim so from their parents' basement.[b] If these had been independent implementations, then there would be reason to worry about the security implications, but they are not.

In a few cases, optimized or license-sanitized versions have been written, but overwhelmingly this is just pointless copy-and-paste of identical source code in blatant disregard of Occam's three-quarters-millennia-old advice.

I am a card-carrying member of the "aghast" segment. My membership card is a FreeBSD commit message shown in the figure here.

My libmd, which is as unencumbered by copyright issues as it can be, later grew more cryptographic hash algorithms, such as RIPEMD-160 and the SHA family, and it has been adopted by some other operating systems.

I am also in the "sigh" segment, because not all mainstream operating systems have adopted libmd, despite having 16 years to do so, and if they have, they do not agree what should be in it. For example, Solaris seems to leave MD2 out (see http://hub.opensolaris.org/bin/view/Project+crypto/libmd), which begs the question: Which part of "software portability" don't they understand?

I am, sadly, also in the "huh?" segment, because there seems to be no hope. The rational thing to expect would be that somebody from The Open Group reads this article, reproduces my statistics, and decides that yes, there is indeed demand for a "libstdcrypto" filled with the usual bunch of crypto algorithms. That, I am told, is impossible. The Open Group does not write new standards; they just bicker over the usability of `${.CURDIR}` in `make(1)` and probably also the potential market demand for fire that can be applied nasally.

The other possible avenue of hope is that the ISO-C standardization group would address this embarrassing situation. Before getting your hopes too high, bear in mind they have still not managed to provide for specification of integer endianness, even though CPUs can do it and hardware and protocols have needed it since the days of the ARPANET.

If the ISO-C crew decided to do it, their process for doing so would undoubtedly consume 5–10 years before a document came out at the other end, by which time SHA-3 would likely be ready, rendering the standard instantly obsolete.

But it is all a pipe dream, if ISO is still allergic to standards with ITAR restrictions. And you can forget everything about a benevolent dictator laying down the wise word as law: Linus doesn't do userland.

To be honest, what I have identified here is probably the absolutely worst-case example.

First, if you need SHA-2, you need SHA-2, and it has to do the right and correct thing for SHA-2. There is little or no room for creativity or improvements, apart from performance.

Second, crypto algorithms are everywhere these days. Practically all communication methods, from good old email over VPNs (virtual private networks) and torrent sites to VoIP (voice over IP), offers strong crypto.

But aren't those exactly the same two reasons why we should not be in this mess to begin with? C

b The fact that MD5 seems to be more in demand—yes, I may indeed be to blame for that myself, but that is a story for another day; search for "md5crypt" if you cannot wait—than its quality warrants is a matter of choice of algorithm, not a matter of implementation of the algorithm chosen.

A card-carrying member of the "aghast" segment.

```
src/lib/libmd/Makefile:

r1802 | phk | 1994-07-24 03:29:56 +0000 (Sun, 24 Jul 1994)


Imported libmd. This library contains MD2, MD4, and MD5.

These three boggers pop up all over the place all of the time, so I

decided we needed a library with them. In general, they are used for

security checks, so if you use them you want to link them static.
```

Related articles on queue.acm.org

Languages, Levels, Libraries, and Longevity
John R. Mashey
http://queue.acm.org/detail.cfm?id=1039532

Gardening Tips
Kode Vicious
http://queue.acm.org/detail.cfm?id=1870147

Poul-Henning Kamp (phk@FreeBSD.org) has programmed computers for 26 years and is the inspiration behind bikeshed.org. His software has been widely adopted as "under the hood" building blocks in both open source and commercial products. His most recent project is the Varnish HTTP accelerator, which is used to speed up large Web sites such as Facebook.

October 22–27, 2011

Co-located with SPLASH/OOPSLA

Hilton Portland & Executive Tower

Portland, Oregon USA

ONWARD! 2011

ACM Conference on New Ideas in Programming and Reflections on Software

Submissions for papers, workshops, essays, and films >> April 8, 2011

Chair
Robert Hirschfeld
Hasso-Plattner-Institut Potsdam, Germany
chair@onward-conference.org

Papers
Eelco Visser
Delft University of Technology, The Netherlands
papers@onward-conference.org

Workshops
Pascal Costanza
Vrije Universiteit Brussel, Belgium
workshops@onward-conference.org

Essays
David West
New Mexico Highlands University, USA
essays@onward-conference.org

Films
Bernd Bruegge
Technische Universität München, Germany
films@onward-conference.org

Association for Computing Machinery

http://onward-conference.org/

DOI:10.1145/1897852.1897871

Compose "dream tools" from continuously evolving bundles of software to make sense of complex scientific data sets.

BY KATY BÖRNER

Plug-and-Play Macroscopes

DECISION MAKING IN science, industry, and politics, as well as in daily life, requires that we make sense of data sets representing the structure and dynamics of complex systems. Analysis, navigation, and management of these continuously evolving data sets require a new kind of data-analysis and visualization tool we call a macroscope (from the Greek macros, or "great," and skopein, or "to observe") inspired by de Rosnay's futurist science writings.[8]

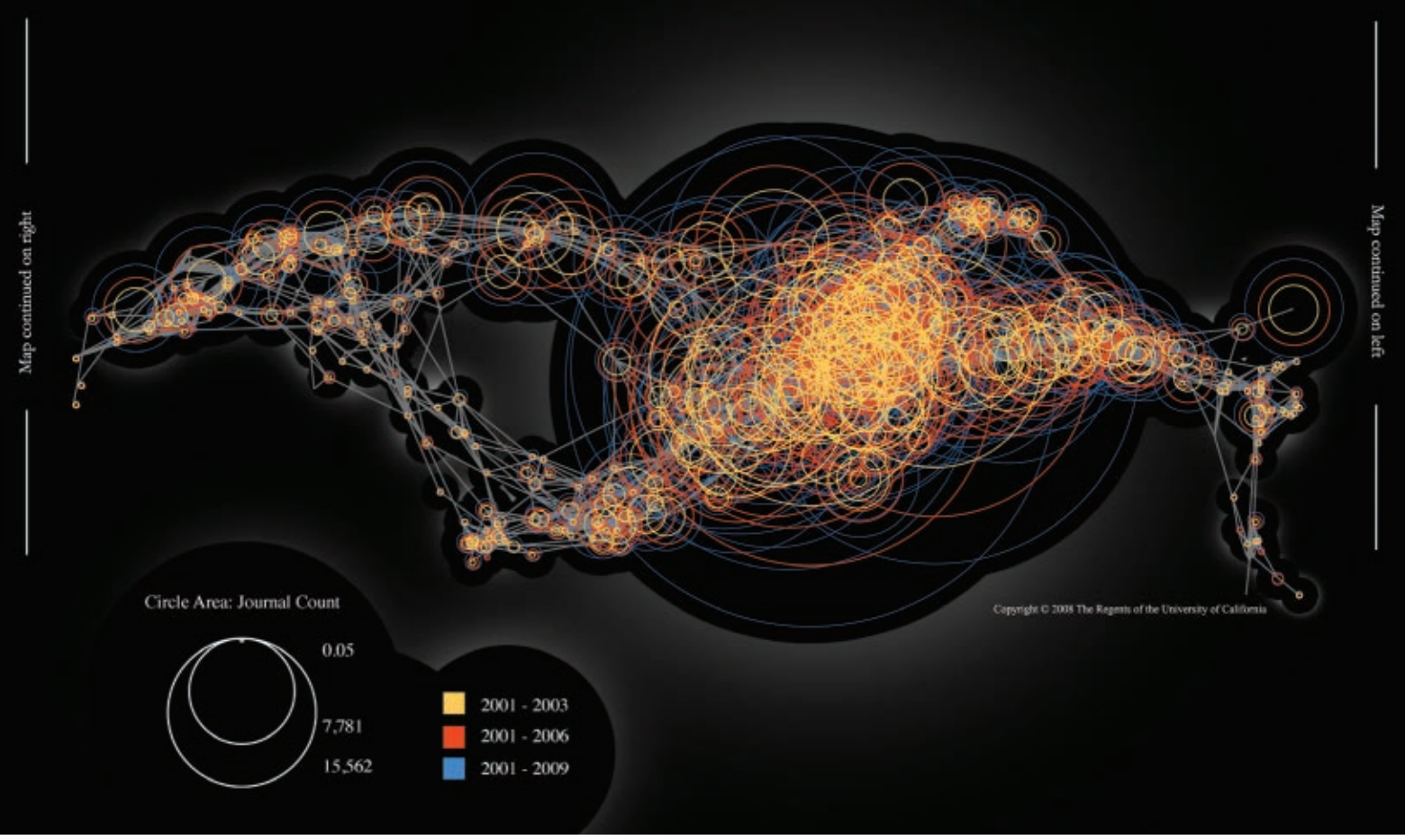
Just as the microscope made it possible for the naked human eye to see cells, microbes, and viruses, thereby advancing biology and medicine, and just as the telescope opened the human mind to the immensity of the cosmos and the conquest of space—the macroscope promises to help make sense of yet another dimension—the infinitely complex. Macroscopes provide a "vision of the whole," helping us "synthesize" the related elements and detect patterns, trends, and outliers while granting access to myriad details.[18,19] Rather than make things larger or smaller, macroscopes let us observe what is at once too great, slow, or complex for the human eye and mind to notice and comprehend.

Many of the best micro-, tele-, and macroscopes are designed by scientists keen to observe and comprehend what no one has seen or understood before. Galileo Galilei (1564–1642) recognized the potential of a spyglass for the study of the heavens, ground and polished his own lenses, and used the improved optical instruments to make discoveries like the moons of Jupiter, providing quantitative evidence for the Copernican theory. Today, scientists repurpose, extend, and invent new hardware and software to create macroscopes that may solve both local and global challenges[20] (see the sidebar "Changing Scientific Landscape").

My aim here is to inspire computer scientists to implement software frameworks that empower domain scientists to assemble their own continuously evolving macroscopes, adding and upgrading existing (and removing obsolete) plug-ins to arrive at a set that is truly relevant for their work—with little or no help from computer scientists. Some macroscopes may resemble cyberinfrastructures (CIs),[1] providing user-friendly access to massive amounts of data, services, computing resources, and expert communities. Others may be Web services or standalone tools. While microscopes and telescopes are physical instruments, macroscopes resemble continuously changing bundles of software plug-ins. Macroscopes make it easy to select and combine algorithm and tool plug-ins but also interface plug-ins, workflow support, logging, scheduling, and other plug-ins needed for scientifically rigorous work. They make it easy to share

» key insights

- **OSGi/CIShell-powered tools improve decision making in e-science, government, industry, and education.**
- **Non-programmers can use OSGi/CIShell to assemble custom "dream tools."**
- **New plug-ins are retrieved automatically via OSGi update services or shared via email and added manually; they can be plugged and played dynamically, without restarting the tool.**



UCSD Map of Science with data overlays of MEDLINE publications that acknowledge NIH funding.

plug-ins via email, flash drives, or online. To use new plug-ins, simply copy the files into the plug-in directory, and they appear in the tool menu ready for use. No restart of the tool is necessary. Sharing algorithm components, tools, and novel interfaces becomes as easy as sharing images on Flickr or videos on YouTube. Assembling custom tools is as quick as compiling your custom music collection.

The macroscopes presented here were built using the Open Services Gateway Initiative Framework (OSGi) industry standard and the Cyberinfrastructure Shell (CIShell) that supports integration of new and existing algorithms into simple yet powerful tools. As of January 2011, six different research communities were benefitting from OSGi and/or CIShell powered tools. Several other tool-development efforts consider adoption.

Related Work

Diverse commercial and academic efforts support code sharing; here, I discuss those most relevant for the design and deployment of plug-and-play macroscopes:

Google Code and SourceForge.net provide the means to develop and distribute software; for example, in August 2009, SourceForge.net hosted more than 230,000 software projects by two million registered users (285,957 in January 2011); also in August 2009 ProgrammableWeb.com hosted 1,366 application programming interfaces and 4,092 mashups (2,699 APIs and 5,493 mashups in January 2011) that combine data or functionality from two or more sources to arrive at a service.

Web services convert any Web browser into a universal canvas for information and service delivery. In addition, there are diverse e-science infrastructures supporting researchers in the composition and execution of analysis and/or visualization pipelines or workflows. Among them are several cyberinfrastructures serving large biomedical communities: the cancer Biomedical Informatics Grid (caBIG) (http://cabig.nci.nih.gov); the Biomedical Informatics Research Network (BIRN) (http://nbirn.net); and the Informatics for Integrating Biology and the Bedside (i2b2) (https://www.i2b2.org). The HUBzero (http://hubzero.org) platform for scientific collaboration uses the Rapture toolkit to serve Java applets, employing the TeraGrid, the Open Science Grid, and other national grid-computing resources for extra cycles. The collaborative environment of myExperiment (http://myexperiment.org) (discussed later) supports the sharing of scientific workflows and other research objects.

Missing so far is a common standard for the design of modular, compatible algorithm and tool plug-ins (also called modules or components) easily combined into scientific workflows (also called pipeline and composition). This leads to duplication of work, as even in the same project, different teams might develop several incompatible "plug-ins" that have almost identical functionality yet are incompatible. Plus, adding a new algorithm plug-in to an existing cyberinfrastructure or bundling and deploying a subset of plug-ins as a new tool/service requires extensive programming skills. Consequently, many innovative new algorithms are never integrated into common CIs and tools due to resource limitations.

Web sites like IBM's Many Eyes (http://manyeyes.alphaworks.ibm.com/manyeyes/visualizations) and Swivel (http://swivel.com) demonstrate the power of community data sharing and visualization. In 2009 alone, Many Eyes

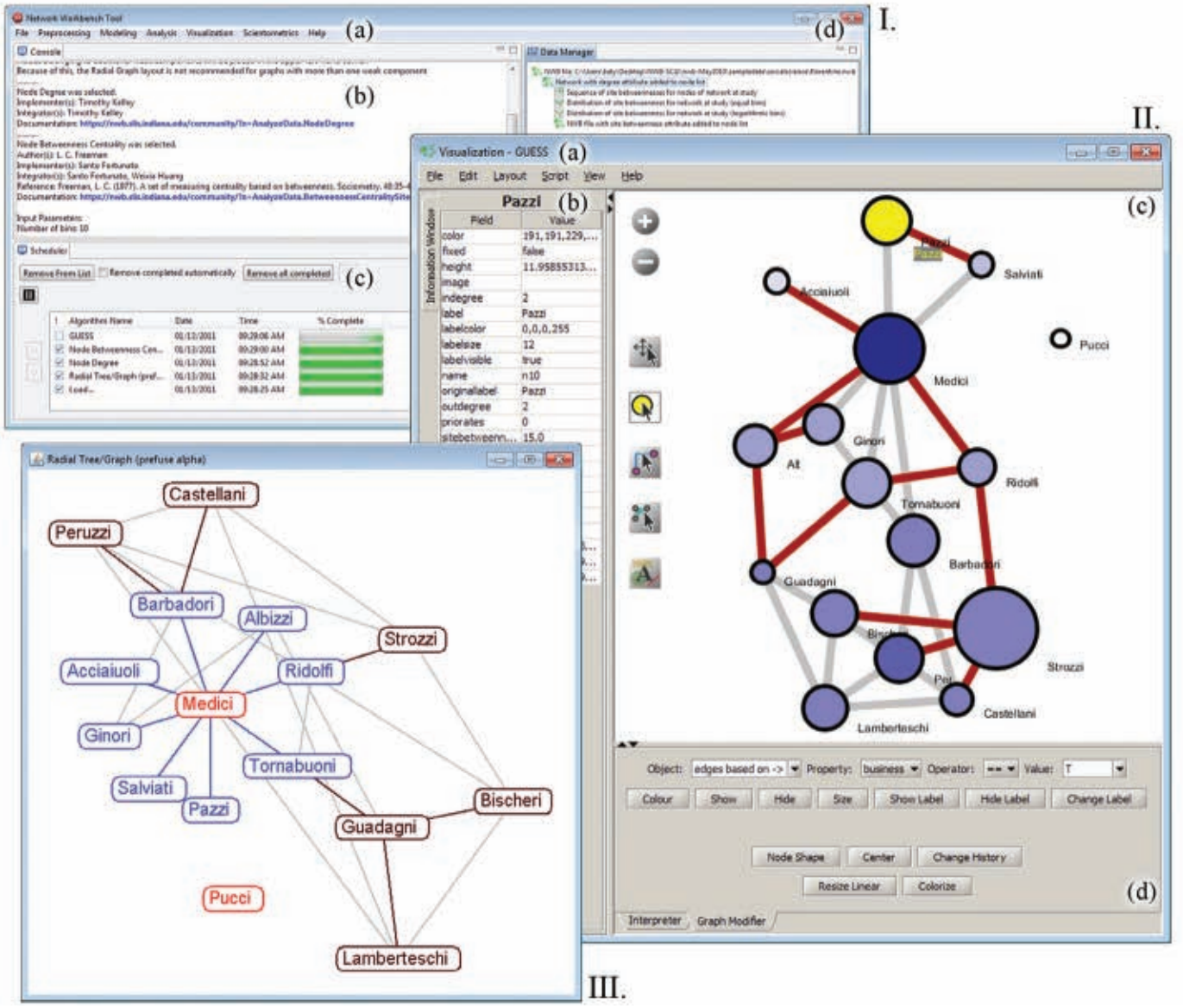


Figure 1. The NWB tool interface (I) with menu (a), *Console* (b), *Scheduler* (c), and *Data Manager* (d). The two visualizations of Renaissance Florentine families used the GUESS tool plug-in (II) and prefuse.org algorithm plug-in (III) Nodes denote families labeled by name; links represent marriage and business relationships among families. In GUESS, nodes are size-coded by wealth and color-coded by degree; marriage relationships are in red using the *Graph Modifier* (d). The "Pazzi" family in (c) was selected to examine properties in the Information Window (b).

had more than 66,429 data sets and 35,842 visualizations, while Swivel had 14,622 data sets and 1,949,355 graphs contributed and designed by 12,144 users. Both sites let users share data (not algorithms), generate and save different visualization types, and provide community support. In January 2011, the numbers for Many Eyes increased to 165,124 data sets and 79,115 visualizations, while Swivel ceased to exist.

Data analysis and visualization is also supported by commercial tools like Tableau (http://tableausoftware.com), Spotfire (http://spotfire.tibco.com), and free tools; see Börner et al.[6] for a review of 20 tools and APIs. While they offer valuable functionality and are widely used in research, education, and industry, none makes it easy for users to share and bundle their algorithms into custom macroscopes.

Plug-and-Play Software Architectures

When discussing software architectures for plug-and-play macroscopes, it is beneficial to distinguish among: (1) the "core architecture" facilitating the plug-and-play of data sets and algorithms; (2) the "dynamic filling" of this core comprising the actual algorithm, tool, user interface, and other plug-ins; (3) and the bundling of all components into "custom tools." To make all three parts work properly, it is important to understand who takes ownership of which ones and what general features are desirable (see the sidebar "Desirable Features and Key Decisions").

Core architecture. To serve the needs of scientists (see both sidebars) the core architecture must empower non-programmers to plug, play, and share their algorithms and to design custom macroscopes and other tools. The solution proposed here is based on OSGi/CIShell:

Open Services Gateway Initiative. Developed by the OSGi Alliance (http://osgi.org), this service platform has

been used since 1999 in industry, including by Deutsche Telekom, Hitachi, IBM, Mitsubishi Electric, NEC, NTT, Oracle, Red Hat, SAP AG, and Siemens Enterprise Communications. It is a dynamic module system for Java, supporting interoperability of applications and services in a mature and comprehensive way with an effective yet efficient API. The platform is interface-based, easing plug-and-play integration of independent components by managing class and dependency issues when combining components. It is also dynamic; that is, new components can be added without stopping the program. It also comes with a built-in mechanism for retrieving new components through the Internet. As service-oriented architecture, OSGi is an easy way to bundle and pipeline algorithms into "algorithm clouds." A detailed description of the OSGi specification and existing reference implementations is beyond the scope of this article but can be explored through http://www.osgi.org/Specifications.

Leveraging OSGi provides access to a large amount of industry-standard code—prebuilt, pretested, continuously updated components—and know-how that would otherwise take years to reinvent/re-implement, thus helping reduce time to market, development, and cost of maintenance. OSGi bundles can be developed and run using a number of frameworks, including the Equinox project from Eclipse (http://eclipse.org/equinox), the reference implementation of the OSGi R4 core framework. Eclipse includes extensive add-ons for writing and debugging code, interacting with code repositories, bug tracking, and software profiling that greatly extend the base platform.

Cyberinfrastructure Shell (http://cishell.org). This open-source software specification adds "sockets" to OSGi into which data sets, algorithms, and tools can be plugged using a wizard-driven process.[11] CIShell serves as a central controller for managing data sets and seamlessly exchanging data and parameters among various implementations of algorithms. It also defines a set of generic data-model APIs and persistence APIs. Extending the data-model APIs makes it possible to implement and integrate various data-model plug-ins. Each data model requires a "persister" plug-in to load, view, and save a data set from/to a data file in a specific format. Some data models lack a persister plug-in, instead converting data to or from some other data format that does have one. CIShell also defines a set of algorithm APIs that allows developers to develop and integrate diverse new or existing algorithms as plug-ins.

Though written in Java, CIShell supports integration of algorithms written in other programming languages, including C, C++, and Fortran. In practice, a pre-compiled algorithm must be wrapped as a plug-in that implements basic interfaces defined in the CIShell Core APIs. Pre-compiled algorithms can be integrated with CIShell by providing metadata about their input and output. Various templates are available for facilitating integration of algorithms into CIShell. A plug-in developer simply fills out a sequence of forms for creating a plug-in and exports it to the installation directory and the new algorithm appears in the CIShell graphical user interface (GUI) menu. This way, any algorithm or tool that can be executed from a command line is easily converted into a CIShell compatible plug-in.

CIShell's reference implementation also includes a number of basic services, including a default menu-driven interface, work-log-tracking module, a data manager, and a scheduler (see Figure 1, left). Work logs—displayed in a console and saved in log files—comprise all algorithm calls and parameters used, references to original papers and online documentation, data loaded or simulated, and any errors. The algorithm scheduler shows all currently scheduled or running processes, along with their progress. CIShell can be deployed as a standalone tool or made available as either a Web or peer-to-peer service. The CIShell Algorithm Developer's Guide[7] details how to develop and integrate Java and non-Java algorithms or third-party libraries.

OSGi/CIShell combined. Software designed using OSGi/CIShell is mainly a set of Java Archive bundles, also called plug-ins. OSGi services, CIShell services, and data set/algorithm services all run in the OSGi container. The CIShell framework API is itself an OSGi bundle that does not register OSGi services, providing instead interfaces for data-set and algorithm services, basic services (such as logging and conversion), and application services (such as scheduler and data manager). Each bundle includes a manifest file with a dependency list stating which packages and other bundles it must run; all bundles are prioritized. Upon application start-up, the bundles with highest priority start first, followed by bundles of second, third, fourth,... priority. Bundles can also be started at runtime.

A bundle can create and register an object with the OSGi service registry under one or more interfaces. The services layer connects bundles dynamically by offering a "publish-find-bind" model for Java objects. Each service registration has a set of standard and custom properties. An expressive filter language is available to select relevant services. Services are dynamic; that is, bundles can be installed and uninstalled on the fly, while other bundles adapt, and the service registry accepts any object as a service. However, registering objects under (standard) interfaces (such as OSGi and CIShell) helps ensure reuse. Due to the declarative specification of bundle metadata, a distributed version of CIShell could be built without changing most algorithms.

The result is that domain scientists can mix and match data sets and algorithms, even adding them dynamically to their favorite tool. All plug-ins that agree on the CIShell interfaces can be run in software designed with the OSGi/CIShell core architecture. No common central data format is needed. Plug-ins can be shared in a flexible, decentralized fashion.

Dynamic filling. As of January 2011, the OSGi/CIShell plug-in pool included more than 230 plug-ins, including approximately 60 "core" OSGi/CIShell plug-ins and a "filling" of more than 170 algorithm plug-ins, plus 40 sample data sets, as well as configuration files and sample data files. Nearly 85% of the algorithm plug-ins are implemented in Java, 5% in Fortran, and the other 10% in C, C++, Jython, and OCaml; see http://cishell.wiki.cns.iu.edu.

Custom tools. The OSGi/CIShell framework is at the core of six plug-and-play tools that resemble simple macroscopes and serve different sci-

entific communities; for example, the Information Visualization Cyberifrastructure (IVC) was developed for research and education in information visualization; the Network Workbench (NWB) tool was designed for large-scale network analysis, modeling, and visualization; the Science of Science (Sci[2]) tool is used by science-of-science researchers, as well as by science-policy analysts; the Epidemics (EpiC) tool is being developed for epidemiologists; TEXTrend supports analysis of text; and DynaNets will be used to advance theory on network diffusion processes. Here, NWB and Sci[2] are covered in detail:

The NWB tool (http://nwb.cns.iu.edu) supports the study of static and dynamic networks in biomedicine, physics, social science, and other research areas. It uses 39 OSGi plug-ins and 18 CIShell plug-ins as its core architecture; two of them define the functionality of the simple GUI in Figure 1 (I), top left with the menu (I.a) for users to load data and run algorithms and tools. The *Console* (I.b) logs all data and algorithm operations, listing acknowledgment information on authors, programmers, and documentation URLs for each algorithm. The Data Manager (I.d) displays all currently loaded and available data sets. A Scheduler (I.c) lets users keep track of the progress of running algorithms. Worth noting is that the interface is easily branded or even replaced (such as with a command-line interface).

The NWB tool includes 21 converter plug-ins that help load data into in-memory objects or into formats the algorithms read behind the scenes. Most relevant for users are the algorithm plug-ins that can be divided into algorithms for preprocessing (19), analysis (56), modeling (10), and visualization (19). Three standalone tools—Discrete Network Dynamics (DND), GUESS, and GnuPlot—are available via the NWB menu system. GUESS is an exploratory data-analysis-and-visualization tool for graphs and networks (http://graphexploration.cond.org), as shown in Figure 1, II, containing a domain-specific embedded language called Gython (an extension of Python, or more specifically Jython) that supports the customization of graph designs. GnuPlot is a portable, command-line-driven, interactive plotting utility for data and related functions (http://gnuplot.info). NWB uses 15 supporting libraries, including Colt, JUNG, Jython, and Prefuse (see Prefuse layouts in Figure 1, III); detailed listings are provided in the NWB tutorial[3] and wiki (http://nwb.wiki.cns.iu.edu).

A common network-science workflow includes data loading and/or modeling, preprocessing, analysis, visualization, and export of results (such as tables, plots, and images). More than 10 different algorithms may be run in one workflow, not counting data converters. Common workflows and references to peer-reviewed papers are given in Börner et al.[3] Here are six exemplary NWB workflows from different application domains:

- Error-tolerance and attack-tolerance analysis in physics and computer science requires loading or modeling a network and deleting random nodes (such as by error) or deleting highly connected hub nodes (such as in an attack);
- Peer-to-peer network analysis in computer science can include simulation of various networks and an analysis of their properties;
- Temporal text analysis in linguistics, information science, and computer science might apply the burst-detection algorithm to identify a sudden increase in the usage frequency of words, with results visualized;
- Social-network analysis in social science, sociology, and scientometrics might compare properties of scholarly and friendship networks for the same set of people; the scholarly network can be derived from publications and the friendship network from data acquired via questionnaires;
- Discrete network dynamics (biology) can be studied through the DND tool, which bundles loading and modeling a multistate discrete network model, to generate the model's state-space graph, analyze the attractors of the state space, and generate a visualization of an attractor basin; and
- Data conversion across sciences can use multiple converter algorithms to translate among more than 20 data formats.

Most workflows require serial application of algorithms developed in different areas of science and contributed by different users. Much of the related complexity is hidden; for example, users do not see how many converters are involved in workflow execution. Only those algorithms that can be applied to a currently selected data set can be selected and run, with all others grayed out. Expert-workflow templates and tutorials provide guidance through the vast space of possible algorithm combinations.

The Science of Science tool (http://sci2.cns.iu.edu). The Sci[2] tool supports the study of science itself through scientific methods; science-of-science studies are also known as scientometric, bibliometric, or informetric studies. Research in social science, political science, physics, economics, and other areas further increases our understanding of the structure and dynamics of science.[2,5,16] The tool supports the study of science at micro (individual), meso (institution, state), and global (all science, international) levels using temporal, geospatial, topical, network-analyses, and visualization techniques (http://sci2.wiki.cns.iu.edu).

Algorithms needed for these analyses are developed in diverse areas of science; for example, temporal-analysis algorithms come from statistics and computer science; geospatial-analysis algorithms from geography and cartography; semantic-analysis algorithms from cognitive science, linguistics, and machine learning; and network analysis and modeling from social science, physics, economics, Internet studies, and epidemiology. These areas have highly contrasting preferences for data formats, programming languages, and software licenses, yet the Sci[2] tool presents them all through a single common interface thanks to its OSGi/CIShell core. Moreover, new algorithms are added easily; in order to read a novel data format, only one new converter must be implemented to convert the new format into an existing format.

Multiple workflows involve more data converters than algorithms, as multiple converters are needed to bridge output and input formats used by consecutive algorithms. Workflows are frequently rerun several times due to imperfect input data, to optimize parameter settings, or to compare different algorithms. Thanks to the Sci[2] tool, an analysis that once required weeks

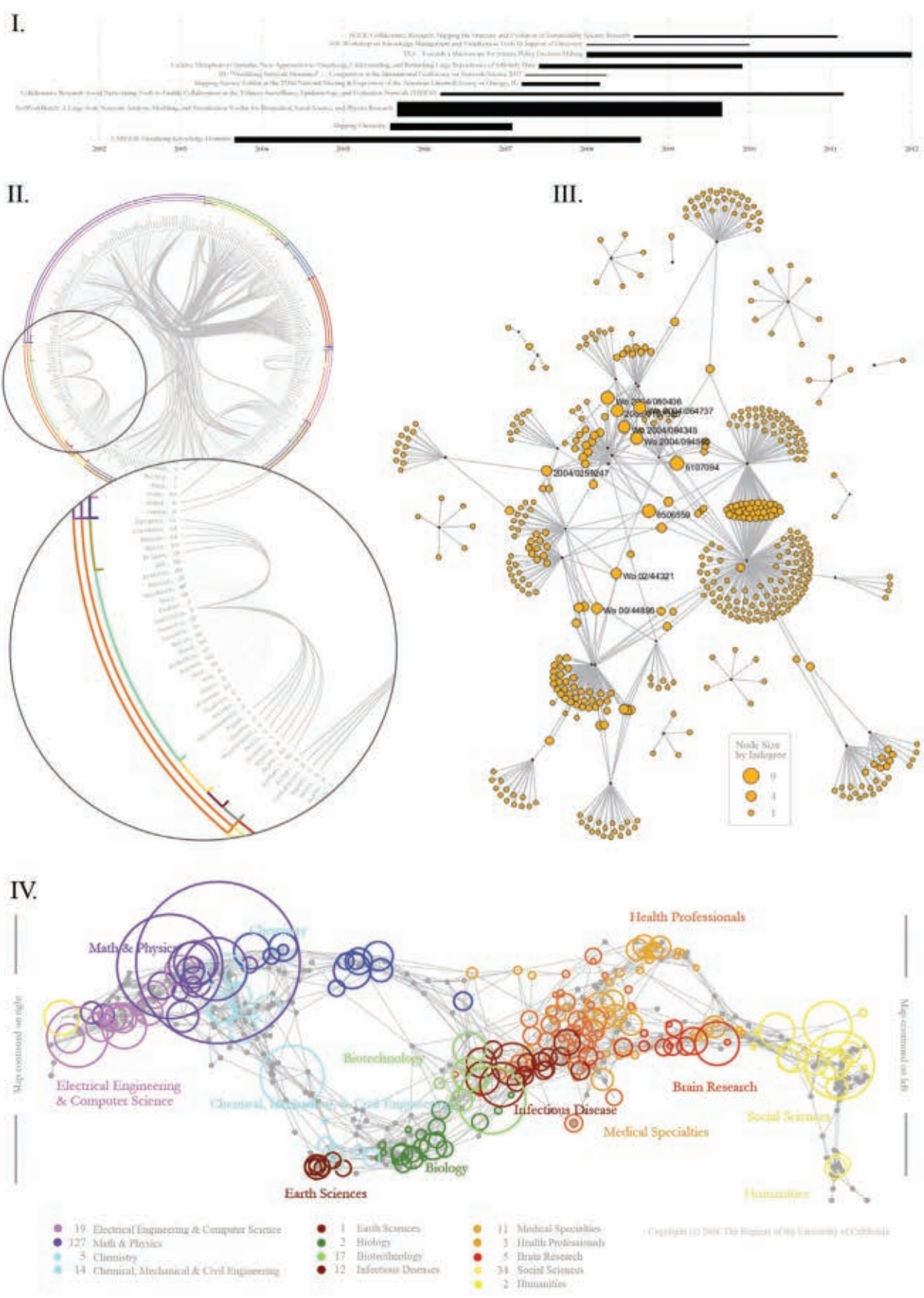


Figure 2. Exemplary Sci² tool workflows: horizontal-bar-graph visualization of NSF funding for one investigator (I); circular layout of a hierarchically clustered co-author network of network-science researchers, with zoom into Eugene Garfield's network (II); citation network of U.S. patents on RNAi and patents they cite, with highly cited patents labeled (III); and UCSD science base map with overlay of publications by network-science researchers (IV).

Changing Scientific Landscape

As science becomes increasingly data driven and computational, as well as collaborative and interdisciplinary, there is increased demand for tools that are easy to extend, share, and customize:

- ***Star scientist → Research teams.*** **Traditionally, science was driven by key scientists. Today, science is driven by collaborating co-author teams, often comprising experts from multiple disciplines and geospatial locations[5,17];**
- ***Users → Contributors.*** **Web 2.0 technologies empower users to contribute to Wikipedia and exchange images, videos, and code via Fickr, YouTube, and SourceForge.net. Wikispecies, WikiProfessionals, and WikiProteins combine wiki and semantic technology to support real-time community annotation of scientific data sets[14];**
- ***Disciplinary → Cross-disciplinary.*** **The best tools frequently borrow and synergistically combine methods and techniques from different disciplines of science, empowering interdisciplinary and/or international teams of researchers, practitioners, and educators to collectively fine-tune and interpret results;**
- ***Single specimen → Data streams.*** **Microscopes and telescopes were originally used to study one specimen at a time. Today, many researchers must make sense of massive data streams of multiple data types and formats and of different dynamics and origin; and**
- ***Static instrument → Evolving cyberinfrastructure.*** **The importance of hardware instruments that are static and expensive tends to decrease relative to software tools and services that are highly flexible and evolving to meet the needs of different sciences. Some of the most successful tools and services are decentralized, increasing scalability and fault tolerance.**

Good software-development practices make it possible for "a million minds" to design flexible, scalable software that can be used by many:

- ***Modularity.*** **Software modules with well-defined functionality accept contributions from multiple users reduce costs and increase flexibility in tool development, augmentation, and customization;**
- ***Standardization.*** **Standards accelerate development, as existing code is leveraged, helping pool resources, support interoperability, and ease migration from research code to production code and hence the transfer of research results into industry applications and products; and**
- ***Open data and open code.*** **The practice of making data sets and code freely available allows users to check, improve, and repurpose data and code, easing replication of scientific studies.**

or months to set up and run can now be designed and optimized in a few hours. Users can also share, rerun, and improve automatically generated work logs. Workflows designed, validated, and published in peer-reviewed works can be used by science-policy analysts and policymakers alike. As of January 2011, the Sci² tool was being used by the National Science Foundation, the National Institutes of Health, the U.S. Department of Energy, and private foundations adding novel plug-ins and workflows relevant for making decisions involving science policy.

The Sci² tool supports many different analyses and visualizations used to communicate results to a range of stakeholders. Common workflows and references to peer-reviewed papers are given in Börner et al.[3] and the Sci² wiki (http://sci2.wiki.cns.iu.edu). Four sample studies are discussed here and included in Figure 2, I–IV:

▸ Funding portfolios (such as funding received by investigators and institutions, as well as provided by agencies) can be plotted using a horizontal bar graph (HBG); for example, all funding for one researcher was downloaded from the National Science Foundation Award Search site (http://nsf.gov/awardsearch), loaded into Sci², and visualized in HBG, as in Figure 2, I. Each project is represented by a bar starting to the left at a certain state date and ending right at an end date, with bar width representing project duration. Bar-area size encodes a numeric property (here total awarded dollar amount), and equipment grants show as narrow bars of significant height. A label (here project name) is given to the left of the bar. Bars can be color-coded by award type (such as Small Business Innovation Research and Career);

▸ Looking for collaboration patterns among major network-science researchers, the publications of four major researchers were downloaded from the *Web of Science* by Thomson Reuters (http://wokinfo.com). The data was then loaded into Sci², the co-author network extracted, the Blondel community-detection algorithm applied to extract hierarchical clusters of the network, and the result laid out using the Circular Hierarchy visualization, with author names plotted in a circle and connecting lines representing co-author links (see Figure 2, II). Two of the researchers share a combined network, while the others are at the centers of unconnected networks. Also shown is a zoom into Eugene Garfield's network;

▸ To understand what patents exist on the topic of RNA interference (RNAi) and how they built on prior work, data was retrieved from the Scholarly Database (http://sdb.cns.iu.edu).[6] Specifically, a query was run over all text in the U.S. patent data set covering 1976–2010. The U.S. Patent and Trademark Office citation table was downloaded, read into the Sci² tool, the patent-citation network extracted, the "indegree" (number of citations within the set) of all patent nodes calculated, and the network displayed in GUESS (see Figure 2, III). The network represents 37 patents (in red) matching the term RNAi and their and the 487 patents they cite (in orange). Nodes are size-coded by in-degree (number of times a patent is cited); patents with at least five citations are labeled by their patent number. One of the most highly cited is no. 6506559 on "Genetic Inhibition by Double-Stranded RNA"; and

▸ The topical coverage of publication output is revealed using a base map of science (such as the University of California, San Diego map in Figure 2, IV.). The map represents 13 major disciplines of science in a variety of colors, further subdivided into 554 research areas. Papers are matched to research areas via their journal names. Multiple journals are associated with each area, and highly interdisciplinary journals (such as *Nature* and *Science*) are fractionally associated with multiple areas.

Circle size represents number of papers published per research area; the number of publications per discipline given below the map. The knowledge input (such as in terms of read or cited papers) and output (such as in terms of published or funded papers) of an individual, institution, or country can be mapped to indicate core competencies. Most publication output of the four network-science researchers is in physics.

These and many other Sci[2] analyses and corresponding visualizations are highly scalable; thousands of authors, references, and projects can be viewed simultaneously, and visualizations can be saved in vector format for further manipulation.

Macroscope Synergies

Just as the value of the earliest telephones increased in proportion to the number of people using them, plug-and-play macroscopes gain value relative to the increase in their core functionality; numbers of data-set and algorithm plug-ins; and the researchers, educators, and practitioners using and advancing them.

OSGi/CIShell-compliant plug-ins can be shared among tools and projects; for example, network-analysis algorithms implemented for the NWB tool can be shared as Java Archive files through email or other means, saved in the plug-in directory of another tool, and made available for execution in the menu system of that tool. Text-mining algorithms originally developed in TEXTrend (discussed later) can be plugged into the Sci[2] tool to support semantic analysis of scholarly texts. Though National Science Foundation funding for the NWB tool formally ended in 2009, NWB's functionality continues to increase, as plug-ins developed for other tools become available. Even if no project or agency were to fund the OSGi/CIShell core for some time, it would remain functional, due to it being lightweight and easy to maintain. Finally, the true value of OSGi/CIShell is due to the continuously evolving algorithm filling and the "custom tools" continuously developed and shared by domain scientists.

Over the past five years, a number of projects have adopted OSGi (and in two cases, CIShell):

As the functionality of OSGi/CIShell-based software frameworks improves, and as the number and diversity of data-set and algorithm plug-ins increases, so too will the capabilities of custom macroscopes.

- *Cytoscape (http://cytoscape.org).* Led by Trey Ideker at the University of California, San Diego, this open-source bioinformatics software platform enables visualization of molecular-interaction networks, gene-expression profiles, and other state data.[15] Inspired by a workshop on software infrastructures in July 2007 (https://nwb.slis.indiana.edu/events/ivsi2007), Mike Smoot and Bruce W. Herr implemented a proof-of-concept OSGi-based Cytoscape core several months later; OSGi bundles are available at http://chianti.ucsd.edu/svn/core3. Once the new Cytoscape 3.0 core is implemented (projected mid-2011), sharing plug-ins between the NWB tool and Cytoscape will be much easier, thereby extending the functionality and utility of both;
- *Taverna Workbench (http://taverna.org.uk).* Developed by the myGrid team (http://mygrid.org.uk) led by Carol Goble at the University of Manchester, U.K., this suite of free open-source software tools helps design and execute workflows,[12] allowing users to integrate many different software tools, including more than 8,000 Web services from diverse domains, including chemistry, music, and social sciences. The workflows are designed in the Taverna Workbench and can then be run on a Taverna Engine, in the Workbench, on an external server, in a portal, on a computational grid, or on a compute cloud. Raven (a Taverna-specific classloader and registry mechanism) supports an extensible and flexible architecture (with approximately 20 plug-ins) but an implementation using an OSGi framework, with alpha release was scheduled for February 2011. The myExperiment (http://myexperiment.org) social Web site supports the finding and sharing of workflows and provides special support for Taverna workflows[9];
- *MAEviz (https://wiki.ncsa.uiuc.edu/display/MAE/Home).* Managed by Shawn Hampton of the National Center for Supercomputing Applications, this open-source, extensible software platform supports seismic risk assessment based on Mid-America Earthquake Center research in the Consequence-Based Risk Management framework.[10] It also uses the Eclipse Rich Client Platform, including Equinox, a com-

Desirable Features and Key Decisions

The socio-technical design of plug-and-play software architectures involves major decisions based on domain requirements to arrive at powerful tools and services:

- ***Division of labor.*** **The design of a "core architecture" requires extensive computer science expertise and a close collaboration with domain experts. Data-set and algorithm plug-ins—the "filling"—are typically provided by domain experts most invested in the data and knowledgeable about the inner workings and utility of different algorithms. The design of "custom tools" is best performed by domain experts, as only they have the expertise needed to bundle different plug-ins relevant for diverse workflows. Technical manuals on how to use, improve, or extend the "core" need to be compiled by computer scientists, while data-set, algorithm, and tool descriptions are written by domain experts;**
- ***Ease of use.*** **As most plug-in contributions come from domain experts with limited programming skills, non-computer scientists must be able to contribute, share, and use plug-ins without having to write new code. What seems to work well is wizard-driven integration of algorithms and data sets, sharing of plug-ins through email and online sites, deploying plug-ins by adding them to the "plug-in directory," and running them via a menu-driven user interface, as in word-processing systems and Web browsers;**
- ***Core vs. plug-ins.*** **The "core architecture" and the plug-in filling can be implemented as sets of plug-in bundles. Determining whether the graphical user interface, interface menu, scheduler, and data manager should be part of the core or its filling depends on the types of tools and services to be delivered;**
- ***Plug-in content and interfaces.*** **Should a plug-in be a single algorithm or an entire tool? What about data converters needed to make the output of one algorithm compatible with the input of another algorithm? Should they be part of the algorithm plug-in? Should they be packaged separately? What general interfaces are needed to communicate parameter settings, input, and output data? Answers are domain-specific, depending on existing tools and practices and the problems domain experts aim to solve;**
- ***Supported (central) data models.*** **Some tools (such as Cytoscape) use a central data model to which all algorithms conform. Others (such as NWB and Sci²) support many internal data models and provide an extensive set of data converters. The former often speeds execution and visual rendering, and the latter eases integration of new algorithms. In addition, most tools support an extensive set of input and output formats, since a tool that cannot read or write a desired data format is usually of little use by domain experts; and**
- ***Supported platforms.*** **Many domain experts are used to standalone tools (like MS Office and Adobe products) running on a specific operating system. A different deployment (such as Web services) is necessary if the software is to be used via Web interfaces.**

ponent framework based on the OSGi standard (https://wiki.ncsa.uiuc.edu/display/MAE/OSGI+Plug-ins).

▸ *TEXTrend (http://textrend.org).* Led by George Kampis at Eötvös Loránd University, Budapest, Hungary, this E.U.-funded project is developing a framework for flexible integration, configuration, and extension of plug-in-based components in support of natural-language processing, classification/mining, and graph algorithms for analysis of business and governmental text corpuses with an inherently temporal component.[13] In 2009, TEXTrend adopted OSGi/CIShell as its core architecture and has since added a number of plug-ins, including: the Unstructured Information Management Architecture (http://incubator.apache.org/uima); the data-mining, machine-learning, classification, visualization toolset WEKA (http://cs.waikato.ac.nz/ml/weka); Cytoscape; Arff2xgmml converter; R (http://r-project.org) via iGgraph and scripts (http://igraph.sourceforge.net); yEd (http://yworks.com); and the CFinder clique percolation-analysis-and-visualization tool (http://cfinder.org). In addition, TEXTrend extended CIShell's workflow support and now offers Web services to researchers.

▸ *DynaNets (http://www.dynanets.org).* Coordinated by Peter M.A. Sloot at the University of Amsterdam, The Netherlands, DynaNets is an E.U.-funded project for studying and developing a new paradigm of computing that employs complex networks. One related tool under development is Dyneta, which uses OSGi/CIShell as its core to support the study of dynamically evolving networks. The tool is able to generate networks corresponding to different network models, execute a specific event chain on them, and analyze the interplay of network structure and dynamics at runtime. The tool will be used to develop a theory of spreading in networks (such as HIV infections and transmission of drug resistance). An initial set of plug-ins is available at http://egg.science.uva.nl/dynanets/nightly/latest.

As the functionality of OSGi/CIShell-based software frameworks improves and the number and diversity of data-set and algorithm plug-ins increase, so too will the capabilities of custom macroscopes.

Outlook

Instead of working at the Library of Alexandria, the Large Hadron Collider, or any of the world's largest optical telescopes, many researchers have embraced Web 2.0 technology as a way to access and share images and videos, along with data sets, algorithms, and tools. They are learning to navigate, manage, and utilize the massive amounts of new data (streams), tools, services, results, and expertise that become available every moment of every day. Computer scientists can help make this a productive experience by empowering biologists, physicists, social scientists, and others to share, reuse, combine, and extend existing algorithms and tools across disciplinary and geospatial boundaries in support of scientific discovery, product development, and education. Computer scientists will have succeeded in the design of the "core architecture" if they are not needed for the filling or bundling of components into custom tools.

The Cyberinfrastructure for Network Science Center (http://cns.iu.edu) at Indiana University is working on the following OSGi/CIShell core extensions, as well as on more effective means for sharing data sets and algorithms via scholarly markets:

Modularity. The OSGi/CIShell core supports modularity at the algorithm level but not at the visualization level. Like the decomposition of workflows

into algorithm plug-ins, it is algorithmically possible to modularize visualization and interaction design. Future work will focus on developing "visualization layers" supporting selection and combination of reference systems, projections/distortions, graphic designs, clustering/grouping, and interactivity.

Streaming data. The number of data sets that are generated and must be understood in real time is increasing; examples are patient-surveillance data streams and models of epidemics that predict the numbers of susceptible, infected, and recovered individuals in a population over time. EpiC tool development funded by the National Institutes of Health contributes algorithms that read and/or output streams of data tuples, enabling algorithms to emit their results as they run, not only on completion. Data-graph visualizations plot these tuple streams in real time, resizing (shrinking) the temporal axis over time.

Web services. The OSGi/CIShell-based tools discussed here are standalone desktop applications supporting offline work on possibly sensitive data, using a GUI familiar to target users. However, some application domains also benefit from online deployment of macroscopes. While the OSGi specification provides basic support for Web services, CIShell must still be extended to make it easy for domain scientists to design their own macroscope Web services.

Incentive design. Many domain experts have trouble trying to use an evolving set of thousands of possibly relevant data sets compiled for specific studies of inconsistent quality and coverage, saved in diverse formats, and tagged using terminology specific to the original research domains. In addition, thousands of algorithms that support different functionality and diverse input and output formats are written in different languages by students and experts in a range of scientific domains and packaged as algorithms or tools using diverse licenses. More-effective means are needed to help domain experts find the data sets and algorithms most relevant for their work, bundle them into efficient workflows, and relate the results to existing work. Scholarly markets resembling a Web 2.0 version of Craigslist.org can help ease the sharing, navigation, and utilization of scholarly data sets and algorithms, reinforcing reputation mechanisms by, say, providing ways to cite and acknowledge users who share, highlight most downloaded and highest-rated contributions, and offer other means for making data sets, algorithms, workflows, and tutorials part of a valued scholarly record.

Acknowledgments

I would like to thank Micah Linnemeier and Russell J. Duhon for stimulating discussions and extensive comments. Bruce W. Herr II, George Kampis, Gregory J. E. Rawlins, Geoffrey Fox, Shawn Hampton, Carol Goble, Mike Smoot, Yanbo Han, and anonymous reviewers provided valuable input and comments to an earlier draft. I also thank the members of the Cyberinfrastructure for Network Science Center (http://cns.iu.edu), the Network Workbench team (http://nwb.cns.iu.edu), and Science of Science project team (http://sci2.cns.iu.edu) for their contributions toward this work. Software development benefits greatly from the open-source community. Full software credits are distributed with the source, but I especially acknowledge Jython, JUNG, Prefuse, GUESS, GnuPlot, and OSGi, as well as Apache Derby, used in the Sci2 tool. This research is based on work supported by National Science Foundation grants SBE-0738111, IIS-0513650, and IIS-0534909 and National Institutes of Health grants R21DA024259 and 5R01MH079068. Any opinions, findings, and conclusions or recommendations expressed here are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

1. Atkins, D.E., Drogemeier, K.K., Feldman, S.I., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messian, P., Ostriker, J.P., and Wright, M.H. *Revolutionizing Science and Engineering Through Cyberinfrastructure. Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure.* National Science Foundation, Arlington, VA, 2003.
2. Börner, Katy. *Atlas of Science: Visualizing What We Know.* MIT Press, Cambridge, MA, 2010; supplemental material at http://scimaps.org/atlas
3. Börner, K., Barabási, A.-L., Schnell, S., Vespignani, A., Wasserman, S., Wernert, E.A., Balcan, D., Beiró, M., Biberstine, J., Duhon, R.J., Fortunato, S., Herr II, B.W., Hidalgo, C.A., Huang, W.B., Kelley, T., Linnemeier, M.W., McCranie, A., Markines, B., Phillips, P., Ramawat, M., Sabbineni, R., Tank, C., Terkhorn, F., and Thakre, V. *Network Workbench Tool: User Manual 1.0.0.*, 2009; http://nwb.cns.iu.edu/Docs/NWBTool-Manual.pdf
4. Börner, K., Chen, C., and Boyack, K.W. Visualizing knowledge domains. In *Annual Review of Information Science & Technology*, B. Cronin, Ed. Information Today, Inc./American Society for Information Science and Technology, Medford, NJ, 2003, 179–255.
5. Börner, K., Dall'Asta, L., Ke, W., and Vespignani, A. Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams. *Complexity (Special Issue on Understanding Complex Systems) 10*, 4 (Mar./Apr. 2005), 57–67.
6. Börner, K., Huang, W.B., Linnemeier, M., Duhon, R.J., Phillips, P., Ma, N., Zoss, A., Guo, H., and Price, M.A. Rete-Netzwerk-Red: Analyzing and visualizing scholarly networks using the Network Workbench tool. *Scientometrics 83*, 3 (June 2010), 863-876.
7. Cyberinfrastructure for Network Science Center. *Cyberinfrastructure Shell (CIShell) Algorithm Developer's Guide*, 2009; http://cishell.wiki.cns.iu.edu
8. de Rosnay, J. *Le Macroscope: Vers une Vision Globale. Editions du Seuil.* Harper & Row Publishers, Inc., New York, 1975.
9. De Roure, D., Goble, C., and Stevens, R. The design and realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows. *Future-Generation Computer Systems 25* (2009), 561–567.
10. Elnashai, A., Hampton, S., Lee, J.S., McLaren, T., Myers, J. D., Navarro, C., Spencer, B., and Tolbert, N. Architectural overview of MAEviz–HAZTURK. *Journal of Earthquake Engineering 12*, 1 Suppl.2, 01 (2008), 92–99.
11. Herr II, B.W., Huang, W.B., Penumarthy, S., and Börner, K. Designing highly flexible and usable cyberinfrastructures for convergence. In *Progress in Convergence: Technologies for Human Wellbeing*, W.S. Bainbridge and M.C. Roco, Eds. Annals of the New York Academy of Sciences, Boston, 2007, 161–179.
12. Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., and Oinn, T. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research (Web Server Issue) 34*, Suppl. 2 (July 1, 2006), W729–W732.
13. Kampis, G., Gulyas, L., Szaszi, Z., and Szakolczi, Z. Dynamic social networks and the TEXTrend/CIShell framework. Presented at the Conference on Applied Social Network Analysis (University of Zürich, Aug. 27–28). ETH Zürich, Zürich, Switzerland, 2009.
14. Mons, B., Ashburner, M., Chicester, C., Van Mulligen, E., Weeber, M., den Dunnen, J., van Ommen, G.-J., Musen, M., Cockerill, M., Hermjakob, H., Mons, A., Packer, A., Pacheco, R., Lewis, S., Berkeley, A., Melton, W., Barris, N., Wales, J., Mejissen, G., Moeller, E., Roes, P.J., Börner, K., and Bairoch, A. Calling on a million minds for community annotation in WikiProteins. *Genome Biology 9*, 5 (2008), R89.
15. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. Cytoscape: A software environment for integrating models of biomolecular interaction networks. *Genome Research 13*, 11 (2002), 2498–2504.
16. Shiffrin, R. and Börner, K. Mapping knowledge domains. *Proceedings of the National Academy of Sciences 101*, Suppl. 1 (Apr. 2004), 5183–5185.
17. Shneiderman, B. Science 2.0. *Science 319*, 5868 (Mar. 2008), 1349–1350.
18. Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages* (Boulder, CO, Sept. 3–6). IEEE Computer Society, Washington, D.C., 1996, 336–343.
19. Thomas, J.J. and Cook, K.A., Eds. *Illuminating the Path: The Research and Development Agenda for Visual Analytics.* National Visualization and Analytics Center, Richland, WA, 2005; http://nvac.pnl.gov/agenda.stm
20. World Bank and International Monetary Fund. *Global Monitoring Report 2009: A Development Emergency.* The World Bank, Washington, D.C., 2009.

Katy Börner (katy@indiana.edu) is the Victor H. Yngve Professor of Information Science at the School of Library and Information Science, Adjunct Professor at the School of Informatics and Computing, and Founding Director of the Cyberinfrastructure for Network Science Center (http://cns.iu.edu) at Indiana University, Bloomington, IN.

DOI:10.1145/1897852.1897872

Effective countermeasures depend on first understanding how users naturally fall victim to fraudsters.

BY FRANK STAJANO AND PAUL WILSON

Understanding Scam Victims: Seven Principles for Systems Security

FROM A HOLISTIC security engineering point of view, real-world systems are often vulnerable to attack despite being protected by elaborate technical safeguards. The weakest point in any security-strengthened system is usually its human element; an attack is possible because the designers thought only about their strategy for responding to threats, without anticipating how real users would react.

We need to understand how users behave and what traits of that behavior make them vulnerable, then design systems security around them. To gain this knowledge, we examine a variety of scams, distilling some general principles of human behavior that explain why the scams work; we then show how they also apply to broader attacks on computer systems insofar as they involve humans. Awareness of the aspects of human psychology exploited by con artists helps not only the public avoid these particular scams but also security engineers build more robust systems.

Over nine series of the BBC TV documentary *The Real Hustle* (http://www.bbc.co.uk/realhustle/) Paul Wilson and Alexis Conran researched the scams most commonly carried out in Britain and, with Jessica-Jane Clement, replicated hundreds of them on unsuspecting victims while filming the action with hidden cameras. The victims were later debriefed, given their money back, and asked for their consent to publish the footage so others would learn not to fall for the same scams (see the sidebar "Representative Scams" to which we refer throughout the main text.)

The objective of the TV show was to help viewers avoid being ripped off by similar scams. Can security researchers do more? By carefully dissecting dozens of scams, we extracted seven recurring behavioral patterns and related principles exhibited by victims and exploited by hustlers. They are not merely small-scale opportunistic scams (known as "short cons") but in-

» key insights

- **We observed and documented hundreds of frauds, but almost all of them can be reduced to a handful of general principles that explain what victims fall for.**
- **These principles cause vulnerabilities in computer systems but were exploited by fraudsters for centuries before computers were invented and are rooted in human nature.**
- **Users fall prey to these principles not because they are gullible but because they are human. Instead of blaming users, understand that these inherent vulnerabilities exist, then make your system robust despite them.**

herent security vulnerabilities of the human element in any complex system. The security engineer must understand them thoroughly and consider their implications toward computer and system security.

Distraction Principle

While we are distracted by what grabs our interest, hustlers can do anything to us and we won't notice.

The young lady who falls prey to the recruitment scam is so engrossed in her job-finding task that she totally fails to even suspect that the whole agency might be a fraud.

Distraction is at the heart of innumerable fraud scenarios. It is also a fundamental ingredient of most magic performances,[5] which is not surprising if we see such performances as a "benign fraud" for entertainment purposes. Distraction is used in all cases involving sleight of hand, including pickpocketing and the special "throw" found in the Monte.

The very presence of "sexy swindler" Jess among the hustlers owes to Distraction, as well as to Need and Greed (discussed later), since sex is such a fundamental human drive. The 2000 computer worm "ILOVEYOU," which reportedly caused $5 billion–$8 billion damage worldwide, exploited these two principles.

In computing, the well-known tension between security and usability is also related to Distraction. Users care only about what they want to access and are essentially blind to the fact that "the annoying security gobbledygook" is there to protect them. Smart crooks exploit this mismatch to their advantage; a lock that is inconvenient to use is often left open.

Distraction also plays a role in the "419," or Nigerian, scam. The hustler, posing as a Nigerian government officer with access to tens of millions of dollars of dodgy money, wants the mark to help transfer the money out of the country in exchange for a slice of it. When the mark accepts the deal, the hustler demands some amount of advance money to cover expenses. New unexpected expenses come up repeatedly, always with the promise that the money is just about to be transferred. These "convincers" keep the mark focused solely on the huge sum he is promised to receive.

Are only unsophisticated 419 victims gullible? Abagnale[1] showed the Distraction principle works equally well on highly educated CTOs and CIOs. In 1999, he visited a company full of programmers frantically fixing code to avert the Y2K bug. He asked the executives how they found all the program-

PHOTOGRAPH BY COLLIN PARKER

mers and was told "these guys from India" knew computers well and were inexpensive. But, Abagnale thought, any dishonest programmer from an offshore firm fixing Y2K problems could also easily implant a backdoor...

People focused on what they want to do are distracted from the task of protecting themselves. Security engineers who don't understand this principle have already lost the battle.

Social Compliance Principle

Society trains people to not question authority. Hustlers exploit this "suspension of suspiciousness" to make us do what they want.

The jeweler in a jewelry-shop scam gratefully hands over necklace and cash when "policeman" Alex says they're needed as evidence, believing him saying they'll be returned later.

Access control to sensitive databases may involve an exploitable human element. For example, social-engineering-expert Mitnick[7] impersonates a policeman to nothing less than a law-enforcement agency. He builds up credibility and trust by exhibiting knowledge of the lingo, procedures, and phone numbers. He makes the clerk consult the National Crime Information Center database and acquires confidential information about a chosen victim. His insightful observation is that the police and military, far from being a tougher target, are inherently more vulnerable to social engineering as a consequence of their strongly ingrained respect for rank.

Social Compliance is the foundation for phishing. For example our banks, which hold all our money, order us to type our password, and, naturally, we do. It's difficult to fault nontechnical users on this one if they fail to notice the site was only a lookalike. Note the conflict between a bank's security department telling customers "never click on email links" and the marketing department of the same bank sending them clickable email advertisements for new financial products, putting the customers in double jeopardy.

System architects must coherently align incentives and liabilities with overall system goals. If users are expected to perform sanity checks rather than blindly follow orders, then social protocols must allow "challenging the authority"; if, on the contrary, users are expected to obey authority unquestioningly, those with authority must relieve them of liability if they obey a fraudster. The fight against phishing and all other forms of social engineering can never be won unless this principle is understood.

Herd Principle

Even suspicious marks let their guard down when everyone around them appears to share the same risks. Safety in numbers? Not if they're all conspiring against us.

In the Monte, most participants are shills. The whole game is set up to give the mark confidence and make him think: "Yes, the game looks dodgy, but other people are winning money," and "Yes, the game looks difficult, but I did guess where the winning disc was, even if that guy lost." Shills are a key ingredient.

In online auctions, a variety of frauds are possible if bidders are in cahoots with the auctioneer. EBay pioneered a reputation system in which bidders and auctioneers rate each other through public feedback. But fraudsters might boost their reputations through successful transactions with shills. Basic reputation systems are largely ineffective against shills.

In online communities and social networks, multiple aliases created by certain participants to give the impression that others share their opinions are indicated as "sock-puppets." In political elections, introducing fake identities to simulate grass-roots support for a candidate is called "astroturfing." In reputation systems in peer-to-peer networks, as opposed to reputation systems in human communities, multiple entities controlled by the same attacker are called "Sybils." The variety of terms created for different contexts testifies to the wide applicability of the Herd principle to many kinds of multiuser systems.

Dishonesty Principle

Our own inner larceny is what hooks us initially. Thereafter, anything illegal we do will be used against us by fraudsters.

In the Monte, the shills encourage the mark to cheat the operator and even help him do it. Then, having fleeced the mark, the operator pretends to notice the mark's attempt at cheating, using it as a reason for closing the game without giving him a chance to argue.

When hustlers sell stolen goods, the implied message is "It's illegal; that's why you're getting such a good deal," so marks won't go to the police once they discover they've been had. The Dishonesty Principle is at the core of the 419; once a mark realizes it's a scam, calling the police is scary because the mark's part of the deal (essentially money laundering) was in itself illegal and punishable. Several victims have gone bankrupt, and some have even committed suicide, seeing no way out of this tunnel.

The security engineer must be aware of the Dishonesty Principle. A

Principles to which victims respond, as identified by three sets of researchers.

Principle	Cialdini (1985–2009)	Lea et al. (2009)	Stajano-Wilson (2009)
Distraction		~	•
Social Compliance (a.k.a. "Authority")	•	○	○
Herd (a.k.a. "Social Proof")	•		○
Dishonesty			•
Kindness	~		•
Need and Greed (a.k.a. "Visceral Triggers")	~	•	○
Scarcity (related to our "Time")	•	○	~
Commitment and Consistency	•	○	
Reciprocation	•		~

• First identified this principle
○ Also lists this principle
~ Lists a related principle

number of attacks on the system go unreported because the victims won't confess to their "evil" part in the process. When corporate users fall prey to a Trojan horse program purporting to offer, say, free access to porn, they have strong incentives not to cooperate with the forensic investigations of system administrators to avoid the associated stigma, even if the incident affected the security of the whole corporate network. Executives for whom righteousness is not as important as the security of their enterprise might consider reflecting such priorities in the corporate security policy, perhaps by guaranteeing discretion and immunity from "internal prosecution" for victims who cooperate with forensic investigations.

Kindness Principle

People are fundamentally nice and willing to help. Hustlers shamelessly take advantage of it.

This principle is, in some sense, the dual of the Dishonesty Principle, as perfectly demonstrated by the Good Samaritan scam. In it, marks are hustled primarily because they volunteer to help. It is loosely related to Cialdini's Reciprocation Principle (people return favors)[2] but applies even in the absence of a "first move" from the hustler. A variety of scams that propagate through email or social networks involve tear-jerking personal stories or follow disaster news (tsunami, earthquake, hurricane), taking advantage of the generous but naïve recipients following their spontaneous kindness before suspecting anything. Many "social engineering" penetrations of computer systems[7] also rely on victims' innate helpfulness.

Need and Greed Principle

Our needs and desires make us vulnerable. Once hustlers know what we want, they can easily manipulate us.

Loewenstein[4] speaks of "visceral factors such as the cravings associated with drug addiction, drive states (such as hunger, thirst, and sexual desire), moods and emotions, and physical pain." We say "Need and Greed" to refer to this spectrum of human needs and desires—all the stuff we really want, regardless of moral judgement. In the 419 scam, what matters most is not necessarily the mark's greed but his or her personal situation; if the mark is on the verge of bankruptcy, needs major surgery, or is otherwise in dire straits, then questioning the offer of a solution is very difficult. In such cases the mark is not greedy, just depressed and hopeful. If someone prays every day for an answer, an email message from a Nigerian Prince might seem like the heaven-sent solution.

The inclusion of sexual appetite as a fundamental human need justifies, through this principle, the presence of a "sexy swindler" in most scams enacted by "the trio." As noted, the Need and Greed Principle and the Distraction Principle are often connected; victims are distracted by (and toward) that which they desire. This drive is exploited by a vast proportion of fraudulent email messages (such as those involving length enhancers, dates with attractive prospects, viruses, and Trojans, including ILOVEYOU).

An enlightened system administrator once unofficially provided a few gigabytes of soft porn on an intranet server in order to make it unnecessary for local users to go looking for such material on dodgy sites outside the corporate firewall, thereby reducing at the same time connection charges and exposure to malware.

If we want to con someone, all we need to know is what they want, even if it doesn't exist. If security engineers do not understand what users want, and that they want it so badly they'll go to any lengths to get it, then they won't understand what drives users and won't be able to predict their behavior. Engineers always lose against fraudsters who do understand how they can lead their marks. This brings us back to the security/usability trade-off: Lecturing users about disabling ActiveX or Flash or Javascript from untrusted sites is pointless if these software components are required to access what users want or need (such as their online social network site or online banking site or online tax return site). Fraudsters must merely promise some enticing content to enroll users as unwitting accomplices who unlock the doors from inside.

The defense strategy should also include user education; as the *Real Hustle* TV show often says, "If it sounds too good to be true, it probably is."

Time Principle

When under time pressure to make an important choice, we use a different decision strategy, and hustlers steer us toward one involving less reasoning.

In the ring-reward rip-off, the mark is made to believe he must act quickly or lose the opportunity. When caught in such a trap, it's very difficult for people to stop and assess the situation properly.

Unlike the theory of rational choice, that is, that humans take their decision after seeking the optimal solution based on all the available information, Simon[8] suggested that "organisms adapt well enough to 'satisfice'; they do not, in general, 'optimize'."

They may "satisfice," or reach a "good-enough" solution, through simplifying heuristics rather than the complex, reasoned strategies needed for finding the best solution, despite heuristics occasionally failing, as studied by Tversky and Kahneman.[10]

Though hustlers may have never formally studied the psychology of decision making, they intuitively understand the shift. They know that, when forced to take a decision quickly, a mark will not think clearly, acting on impulse according to predictable patterns. So they make their marks an offer they can't refuse, making it clear to them that it's their only chance to accept it. This pattern is evident in the 419 scam and in phishing ("You'll lose access to your bank account if you don't confirm your credentials immediately") but also in various email offers and limited-time discounts in the gray area between acceptable marketing techniques and outright swindle. As modern computerized marketing relies more and more on profiling individual consumers to figure out how to press their buttons, we might periodically have to revise our opinions about which sales methods, while not yet illegal, are ethically acceptable.

From a systems point of view, the Time Principle is particularly important, highlighting that, due to the human element, the system's response to the same stimulus may be radically different depending on the urgency with which it is requested. In military contexts this is taken into account by wrapping dangerous situations that require rapid response (such as challeng-

Representative Scams

Since 2006, the *Real Hustle* TV show has recreated hundreds of scams during which Paul, Alex, and Jess defrauded unsuspecting victims before hidden cameras. Here are five instructive ones:

In the lingo of this peculiar "trade," the victim of the scam is the mark, the perpetrator is the operator, and any accomplice pretending to be a regular customer is a shill.

Monte. **This classic scam involves an operator manipulating three cards (or disks or shells: there are many variations), one of which wins, while the other two lose. The operator shows the player the cards, turns them over face down, then moves them around on the table in full view. Players must follow the moves and put money on the card they believe to be the winner. The operator pays out an equal amount if the player guessed correctly or otherwise pockets the player's money.**

Technically, at the core of the scam is a sleight-of-hand trick whereby the operator undetectably switches two cards. One might therefore imagine the basic scam to consist of performing a few "demo runs" where marks are allowed to guess correctly, then have them bet with real money and at that point send the winning card elsewhere.

But this so-called "game" is really a cleverly structured piece of street theater designed to attract passersby and hook them into the action. The sleight-of-hand element is actually least important; it is the way marks are manipulated, rather than the props, that brings in the money. It's all about the crowd of onlookers and players (all shills) betting in a frenzy and irresistibly sucking marks into wanting a piece of the action.

The Monte is an excellent example that nothing is what it seems, even if the marks think they know what to expect. Many people claim to be able to beat the game, purely because they understand the mechanics of the secret move. But it's impossible to tell whether an experienced operator has made the switch. More important, even if the cards were marked in some way, there is absolutely no way for a legitimate player to secure a win; should a mark consistently bet on the correct position, then other players, actually shills, would over-bet him, "forcing" the operator to take the larger bet. This frustrates the mark, who often increases his bet to avoid being topped. One shill will then pretend to help the mark by bending a corner of the winning card while the operator is distracted, making the mark think he has an unbeatable advantage. This is a very strong play; marks have been seen to drop thousands of dollars only to find the bent card is actually a loser. While mixing the cards, it is possible for a skilled operator to switch the cards and switch the bend from one card to another.

The idea that one can beat the game at all reveals a key misunderstanding—that, in fact, it is not a game in the first place. Monte mobs never pay out to the

From right to left: Paul, with Alex as a shill, scams two marks at the three-shells game (one of several variants of the Monte).

From right to left: Paul and Alex haggle with the mark over the reward in the Ring Reward Rip-off.

Alex, flashing a fake police badge, pretends to arrest Jess in the Jewelry Shop Scam.

ALL IMAGES COURTESY OF OBJECTIVE PRODUCTIONS

ing strangers at a checkpoint or being ordered to launch a nuclear missile) in special "human protocols" meant to enforce, even under time pressure, some of the step-by-step rational checks the heuristic strategy would otherwise omit.

The security architect must identify the situations in which the humans in the system may suddenly be put under time pressure by an attacker and whether the resulting switch in decision strategy might open a vulnerability. This directive applies to anything from retail situations to stock trading and online auctions and from admitting visitors into buildings to handling medical emergencies. Devising a human protocol to guide and pace the response of the potential victim toward the desired goal may be an adequate safeguard and also relieve the victim from stressful responsibility.

Related Work

While a few narrative accounts of scams and frauds are available, from Maurer's study of the criminal world[6] that inspired the 1973 movie *The Sting* to the autobiographical works of notable fraudsters,[1,7] the literature contains little about systematic studies of fraudsters' psychological techniques. But we found two notable exceptions: Cialdini's outstanding book *Influence: Science and Practice*,[2] based on undercover field research, revealed how salespeople's "weapons of influence" are remarkably similar to those of fraudsters; indeed, all of his principles apply to our scenario and vice versa. Meanwhile, Lea et al.[3] examined postal scams, based on a wealth of experimental data, including interviews with victims and lexical analysis of fraudulent letters. Even though our approaches were quite different, our findings are in substantial agreement. The table here summarizes and compares the principles identified in each of these works.

Conclusion

We supported our thesis—that systems involving people can be made secure only if designers understand and acknowledge the inherent vulnerabilities of the "human factor"—with three main contributions:

First is a vast body of original research on scams, initially put together by Wilson and Conran. It started as a TV show, not as a controlled scientific experiment, but our representative write-up[9] still offers valuable firsthand data not otherwise available in the literature;

Second, from these hundreds of scams, we abstracted seven principles. The particular principles are not that important, and others have found

marks; they keep all the money moving between the shills and the operator. The marks are allowed to place a bet only if it's already a loser. Having studied Monte all over the world, we can say it's nothing short of a polite way to mug people.

Ring reward rip-off. The gorgeous Jess buys a cheap ring from a market stall for $5. She then goes to a pub and seductively befriends the barman (the mark). She makes it obvious she's very rich; showing off to her friend (a shill), she makes sure the mark overhears that she just received this amazing $3,500 diamond ring for her birthday. She then leaves.

Paul and Alex arrive at the pub, posing as two blokes having a pint. Jess then phones the pub, very worried, calling her friend the barman by name, saying she lost that very precious ring. Could he check if it's there somewhere? The mark checks, and, luckily, a customer (Paul) found the ring. However, instead of handing it over, Paul demands a reward. The barman gets back to the phone and Jess, very relieved to hear the ring is there, says, without prompting, that she'll give $200 to the person who found it. But the barman goes back to Paul and says the reward is only $20. That's when the hustlers know they've got him; he's trying to make some profit for himself. Paul haggles a bit and eventually returns the ring to the barman for $50. The mark is all too happy to advance the money to Paul, expecting to get much more from Jess. Jess, of course, never calls back.

A convicted criminal proudly says he once made a $2,000 profit with this particular hustle.

Jewelry-shop scam. Jess attempts to buy an expensive necklace but is "arrested" by Alex and Paul posing as plainclothes police officers who expose her as a well-known fraudster, notorious for paying with counterfeit cash. The "cops" collect as evidence the "counterfeit" (actually genuine) cash and, crucially, the necklace, which will, of course, "be returned." The jeweler is extremely grateful the cops saved her from the evil fraudster.

Ironically, as Jess is taken away in handcuffs, the upset jeweler spits out a venomous "Bitch! You could have cost me my job. You know that?"

Recruitment scam. Hustlers set up a fake recruitment agency and, as part of the sign-on procedure, collect all of the applicants' personal details, including mother's maiden name, date of birth, bank-account details, passport number, even PIN—by asking them to protect their data with a four-digit code, as many people memorize only one PIN and use it for everything. With this loot, the hustlers are free to engage in identity theft on everyone who came in for an interview.

Good Samaritan scam. In a parking lot, Jess has jacked up her car but seems stuck. When another car stops nearby, she politely asks the newcomers to help her change the tire, which they do. Apologizing for her cheekiness, she then also asks them if she could get into their car, as she's been out in the cold for a while and is freezing. The gentleman gives her the keys to his car (required to turn on the heat) and, while the marks are busy changing her tire, she drives off with the car. But didn't Jess just lose her original car? No, because it wasn't hers to start with; she just jacked up a random one in the parking lot. To add insult to injury, the marks will also have some explaining to do when the real owners of the car arrive.

A mark, debriefed by accompanying TV crew, is dismayed to learn the hustlers just got hold of all her sensitive personal details in the Recruitment Scam.

From right to left: Jess gets two marks to change her tire before tricking them into handing over their own car keys in the Good Samaritan Scam.

slightly different ones. What matters is recognizing the existence of a small set of behavioral patterns that ordinary people exhibit and that hustlers have been exploiting forever; and

Third, perhaps most significant, we applied the principles to a more general systems point of view. The behavioral patterns are not just opportunities for small-scale hustles but also vulnerabilities of the human component of any complex system.

Our message for the system-security architect is that it is naïve to lay blame on users and whine, "The system I designed would be secure, if only users were less gullible." The wise security designer seeking a robust solution will acknowledge the existence of these vulnerabilities as an unavoidable consequence of human nature and actively build safeguards that prevent their exploitation.

Acknowledgments

Special thanks to Alex Conran for co-writing the TV series and to Alex and Jess Clement for co-starring in it. Thanks to Joe Bonneau, danah boyd, Omar Choudary, Saar Drimer, Jeff Hancock, David Livingstone Smith, Ford-Long Wong, Ross Anderson, Stuart Wray, and especially Roberto Viviani for useful comments on previous drafts. This article is updated and abridged from the 2009 technical report[9] by the same authors. C

References

1. Abagnale, F.W. *The Art of the Steal: How to Protect Yourself and Your Business from Fraud.* Broadway Books, New York, 2001.
2. Cialdini, R.B. *Influence: Science and Practice, Fifth Edition.* Pearson, Boston, MA, 2009; (First Edition 1985).
3. Lea et al. *The Psychology of Scams: Provoking and Committing Errors of Judgement.* Technical Report OFT1070. University of Exeter School of Psychology. Office of Fair Trading, London, U.K., May 2009.
4. Loewenstein, G. Out of control: Visceral influences on behavior. *Organizational Behavior and Human Decision Processes 65*, 3 (Mar. 1996), 272–292.
5. Macknik, S.L., King, M., Randi, J., Robbins, A., Teller, Thompson, J., and Martinez-Conde, S. Attention and awareness in stage magic: Turning tricks into research. *Nature Reviews Neuroscience 9*, 11 (Nov. 2008), 871–879.
6. Maurer, D.W. *The Big Con: The Story of the Confidence Man.* Bobbs-Merrill, New York, 1940.
7. Mitnick, K.D. *The Art of Deception: Controlling the Human Element of Security.* John Wiley & Sons, Inc., New York, 2002.
8. Simon, H.A. Rational choice and the structure of the environment. *Psychological Review 63*, 2 (Mar. 1956), 129–138.
9. Stajano, F. and Wilson, P. *Understanding Scam Victims: Seven Principles for Systems Security.* Technical Report UCAM-CL-TR-754. University of Cambridge Computer Laboratory, Cambridge, U.K, 2009.
10. Tversky, A. and Kahneman, D. Judgment under uncertainty: Heuristics and biases. *Science 185*, 4157 (Sept. 1974), 1124–1131.

Frank Stajano (frank.stajano@cl.cam.ac.uk) is a university senior lecturer in the Computer Laboratory of the University of Cambridge, Cambridge, U.K.

Paul Wilson (info@conartist.tv) is an expert on cheating, award-winning conjuror, and magic inventor. He works in film and television in London and Los Angeles.

DOI:10.1145/1897852.1897873

The advent of multicore processors as the standard computing platform will force major changes in software design.

BY NIR SHAVIT

Data Structures in the Multicore Age

"MULTICORE PROCESSORS ARE about to revolutionize the way we design and use data structures."

You might be skeptical of this statement; after all, are multicore processors not a new class of multiprocessor machines running parallel programs, just as we have been doing for more than a quarter of a century?

The answer is no. The revolution is partly due to changes multicore processors introduce to parallel architectures; but mostly it is the result of the change in the applications that are being parallelized: multicore processors are bringing parallelism to mainstream computing.

Before the introduction of multicore processors, parallelism was largely dedicated to computational problems with regular, slow-changing (or even static) communication and coordination patterns. Such problems arise in scientific computing or in graphics, but rarely in systems.

The future promises us multiple cores on anything from phones to laptops, desktops, and servers, and therefore a plethora of applications characterized by complex, fast-changing interactions and data exchanges.

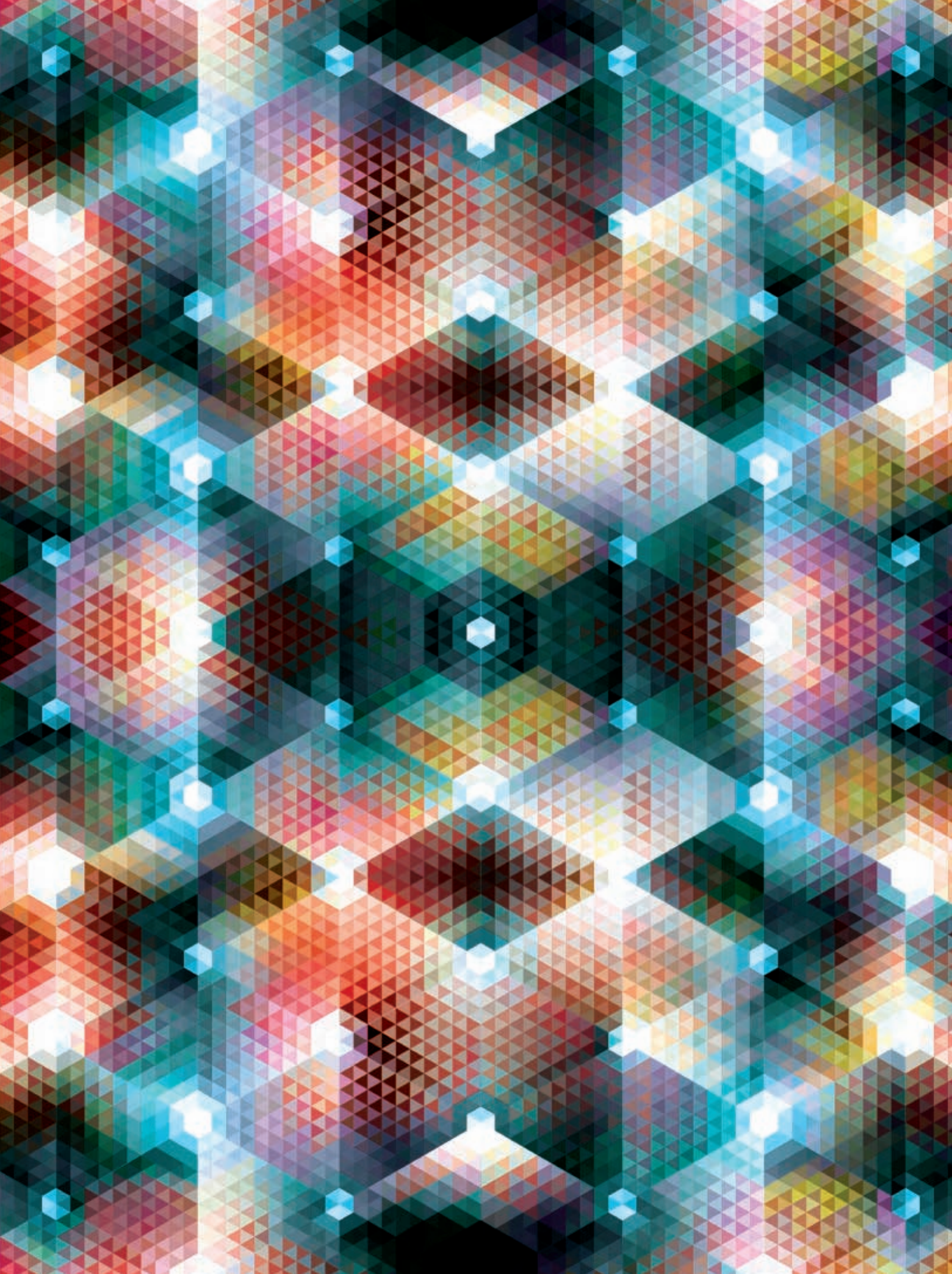
Why are these dynamic interactions and data exchanges a problem? The formula we need in order to answer this question is called *Amdahl's Law*. It captures the idea that the extent to which we can speed up any complex computation is limited by how much of the computation must be executed sequentially.

Define the *speedup S* of a computation to be the ratio between the time it takes one processor to complete the computation (as measured by a wall clock) versus the time it takes n concurrent processors to complete the same computation. Amdahl's Law characterizes the maximum speedup S that can be achieved by n processors collaborating on an application, where p is the fraction of the computation that can be executed in parallel. Assume, for simplicity, that it takes (normalized) time 1 for a single processor to complete the computation. With n concurrent processors, the parallel part takes time p/n, and the sequential part takes time $1-p$. Overall, the parallelized computation takes time $1-p+\frac{p}{n}$. Amdahl's Law says the speedup, that is, the ratio between

» key insights

- **We are experiencing a fundamental shift in the properties required of concurrent data structures and of the algorithms at the core of their implementation.**
- **The data structures of our childhood — stacks, queues, and heaps — will soon disappear, replaced by looser "unordered" concurrent constructs based on distribution and randomization.**
- **Future software engineers will need to learn how to program using these novel structures, understanding their performance benefits and their fairness limitations.**

ILLUSTRATION BY ANDY GILMORE



the sequential (single-processor) time and the parallel time, is:

$$S = \frac{1}{1 - p + \frac{p}{n}}$$

In other words, S does not grow linearly in n. For example, given an application and a 10-processor machine, Amdahl's Law says that even if we manage to parallelize 90% of the application, but not the remaining 10%, then we end up with a fivefold speedup, but not a 10-fold speedup. Doubling the number of cores to 20 will only raise us to a sevenfold speedup. So the remaining 10%, those we continue to execute sequentially, cut our utilization of the 10 processor machine in half, and limit us to a 10-fold speedup no matter how many cores we add.

What are the 10% we found difficult to parallelize? In many mainstream applications they are the parts of the program involving interthread interaction and coordination, which on multicore machines are performed by concurrently accessing shared data structures. Amdahl's Law tells us it is worthwhile to invest an effort to derive as much parallelism as possible from these 10%, and a key step on the way to doing so is to have highly parallel concurrent data structures.

Unfortunately, concurrent data structures are difficult to design. There is a kind of tension between correctness and performance: the more one tries to improve performance, the more difficult it becomes to reason about the resulting algorithm as being correct. Some experts blame the widely accepted threads-and-objects programming model (that is, threads communicating via shared objects), and predict its eventual demise will save us. My experience with the alternatives suggests this model is here to stay, at least for the foreseeable future. So let us, in this article, consider correctness and performance of data structures on multicore machines within the threads-and-objects model.

In the concurrent world, in contrast to the sequential one, correctness has two aspects: safety, guaranteeing that nothing bad happens, and liveness, guaranteeing that eventually something good will happen.

The safety aspects of concurrent data structures are complicated by the need to argue about the many possible interleavings of methods called by different threads. It is infinitely easier and more intuitive for us humans to specify how abstract data structures behave in a sequential setting, where there are no interleavings. Thus, the standard approach to arguing the safety properties of a concurrent data structure is to specify the structure's properties sequentially, and find a way to map its concurrent executions to these "correct" sequential ones. There are various approaches for doing this, called consistency conditions. Some familiar conditions are serializability, linearizability, sequential consistency, and quiescent consistency.

When considering liveness in a concurrent setting, the good thing one expects to happen is that method calls eventually complete. The terms under which liveness can be guaranteed are called progress conditions. Some familiar conditions are deadlock-freedom, starvation-freedom, lock-freedom, and wait-freedom. These conditions capture the properties an implementation requires from the underlying system scheduler in order to guarantee that method calls complete. For example, deadlock-free implementations depend on strong scheduler support, while wait-free ones do all the work themselves and are independent of the scheduler.

Finally, we have the performance of our data structures to consider. Historically, uniprocessors are modeled as Turing machines, and one can argue the theoretical complexity of data structure implementations on uniprocessors by counting the number of steps—the machine instructions—that method calls might take. There is an immediate correlation between the theoretical number of uniprocessor steps and the observed time a method will take.

In the multiprocessor setting, things are not that simple. In addition to the actual steps, one needs to consider whether steps by different threads require a shared resource or not, because these resources have a bounded capacity to handle simultaneous requests. For example, multiple instructions accessing the same location in memory cannot be serviced at the same time. In its simplest form, our theoretical complexity model requires us to consider a new element: stalls.[2,7–10] When threads concurrently access a shared resource, one succeeds and others incur stalls. The overall complexity of the algorithm, and hence the time it might take to complete, is correlated to the number of operations together with the number of stalls (obviously this is a crude model that does not take into account the details of cache coherence). From an algorithmic design point of view, this model introduces a continuum starting from centralized structures where all threads share data by accessing a small set of locations, incurring many stalls, to distributed structures with multiple locations, in which the number of stalls is greatly reduced, yet the number of steps necessary to properly share data and move it around increases significantly.

How will the introduction of multicore architectures affect the design of concurrent data structures? Unlike on uniprocessors, the choice of algorithm will continue, for years to come, to be greatly influenced by the underlying machine's architecture. In particular, this includes the number of cores, their layout with respect to memory and to each other, and the added cost of synchronization instructions (on a multiprocessor, not all steps were created equal).

However, I expect the greatest change we will see is that concurrent data structures will go through a substantiative "relaxation process." As the number of cores grows, in each of the categories mentioned, consistency conditions, liveness conditions, and the level of structural distribution, the requirements placed on the data structures will have to be relaxed in order to support scalability. This will put a burden on programmers, forcing them to understand the minimal conditions their applications require, and then use as relaxed a data structure as possible in the solution. It will also place a burden on data structure designers to deliver highly scalable structures once the requirements are relaxed.

This article is too short to allow a survey of the various classes of concurrent data structures (such a survey can be found in Moir and Shavit[17]) and how one can relax their definitions and implementations in order to make them

scale. Instead, let us focus here on one abstract data structure—a stack—and use it as an example of how the design process might proceed.

I use as a departure point the acceptable sequentially specified notion of a `Stack<T>` object: a collection of items (of type `T`) that provides `push()` and `pop()` methods satisfying the *last-in-first-out* (LIFO) property: the last item pushed is the first to be popped.

We will follow a sequence of refinement steps in the design of concurrent versions of stacks. Each step will expose various design aspects and relax some property of the implementation. My hope is that as we proceed, the reader will grow to appreciate the complexities involved in designing a correct scalable concurrent data-structure.

A Lock-based Stack

We begin with a `LockBasedStack<T>` implementation, whose Java pseudocode appears in figures 1 and 2. The pseudocode structure might seem a bit cumbersome at first, this is done in order to simplify the process of extending it later on.

The lock-based stack consists of a linked list of nodes, each with `value` and `next` fields. A special `top` field points to the first list node or is *null* if the stack is empty. To help simplify the presentation, we will assume it is illegal to add a *null* value to a stack.

Access to the stack is controlled by a single *lock*, and in this particular case a *spin-lock*: a software mechanism in which a collection of competing threads repeatedly attempt to choose exactly one of them to execute a section of code in a mutually exclusive manner. In other words, the winner that acquired the lock proceeds to execute the code, while all the losers spin, waiting for it to be released, so they can attempt to acquire it next.

The lock implementation must enable threads to decide on a winner. This is done using a special synchronization instruction called a `compareAndSet()` (CAS), available in one form or another on all of today's mainstream multicore processors. The CAS operation executes a read operation followed by a write operation, on a given memory location, in one indivisible hardware step. It takes two arguments: an *expected* value and an *update* value. If the memory location's value is equal to the expected value, then it is replaced by the update value, and otherwise the value is left unchanged. The method call returns a Boolean indicating whether the value changed. A typical CAS takes significantly more machine cycles than a read or a write, but luckily, the performance of CAS is improving as new generations of multicore processors role out.

In Figure 1, the `push()` method creates a new node and then calls `tryPush()` to try to acquire the lock. If the CAS is successful, the lock is set to true and the method swings the top reference from the current top-of-stack to its successor, and then releases the lock by setting it back to *false*. Otherwise, the `tryPush()` lock acquisition attempt is repeated. The `pop()` method

Figure 1. A lock-based `Stack<T>`: in the `push()` method, threads alternate between trying to push an item onto the stack and managing contention by backing off before retrying after a failed push attempt.

```
1   public  class  LockBasedStack<T> {
2     private AtomicBoolean lock =
3             new AtomicBoolean(false);
4     ...
5     protected boolean tryPush(Node node) {
6       boolean gotLock = lock.compareAndSet(false, true);
7       if (gotLock)  {
8        Node oldTop = top;
9        node.next = oldTop;
10       top = node;
11       lock.set ( false );
12       }
13    return gotLock;
14    }
15    public void push(T value) {
16      Node node = new Node(value);
17      while (true) {
18       if (tryPush(node)) {
19         return;
20       } else {
21         contentionManager.backoff();
22       }
23    }
24  }
```

Figure 2. The lock-based `Stack<T>`: The `pop()` method alternates between trying to pop and backing off before the next attempt.

```
1   protected Node tryPop() throws EmptyException {
2     boolean gotLock = lock.compareAndSet(false, true);
3     if (gotLock) {
4       Node oldTop = top;
5       if  (oldTop == null) {
6        lock . set ( false );
7        throw new EmptyException();
8       }
9     top = oldTop.next;
10    return oldTop;
11    lock . set ( false );
12    }
13    else return null ;
14  }
15  public T pop() throws EmptyException {
16    while (true) {
17      Node returnNode = tryPop();
18      if  (returnNode != null) {
19       return returnNode.value ;
20      } else {
21        contentionManager.backoff();
22      }
23    }
24  }
```

in Figure 2 calls tryPop(), which attempts to acquire the lock and remove the first node from the stack. If it succeeds, it throws an exception if the stack is empty, and otherwise it returns the node referenced by top. If tryPop() fails to acquire the lock it returns *null* and is called again until it succeeds.

What are the safety, liveness, and performance properties of our implementation? Well, because we use a single lock to protect the structure, it is obvious its behavior is "atomic" (the technical term used for this is *linearizable*[15]). In other words, the outcomes of our concurrent execution are equivalent to those of a sequential execution in which each push or pop take effect at some non-overlapping instant during their method calls. In particular, we could think of them taking effect when the executing thread acquired the lock. Linearizability is a desired property because linearizable objects can be composed without having to know anything about their actual implementation.

But there is a price for this obvious atomicity. The use of a lock introduces a dependency on the operating system: we must assume the scheduler will not involuntarily preempt threads (at least not for long periods) while they are holding the lock. Without such support from the system, all threads accessing the stack will be delayed whenever one is preempted. Modern operating systems can deal with these issues, and will have to become even better at handling them in the future.

In terms of progress, the locking scheme is deadlock-free, that is, if several threads all attempt to acquire the lock, one will succeed. But it is not starvation-free: some thread could be unlucky enough to always fail in its CAS when attempting to acquire the lock.

The centralized nature of the lock-based stack implementation introduces a sequential bottleneck: only one thread at a time can complete the update of the data structure's state. This, Amdahl's Law tells us, will have a very negative effect on scalability, and performance will not improve as the number of cores/threads increases.

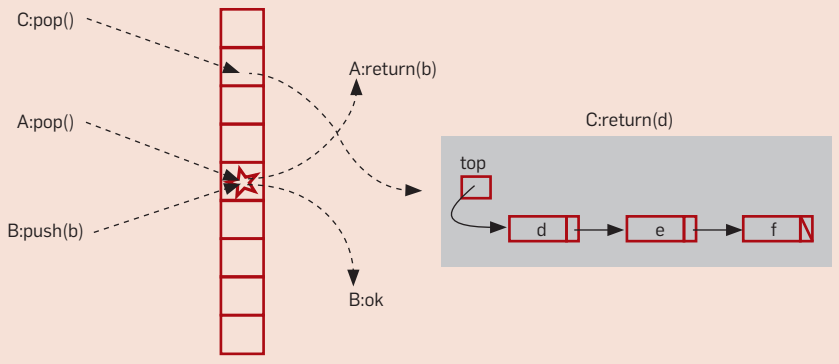
But there is another separate phenomenon here: memory contention. Threads failing their CAS attempts on the lock retry the CAS again even while the lock is still held by the last CAS "winner" updating the stack. These repeated attempts cause increased traffic on the machine's shared bus or interconnect. Since these are bounded resources, the result is an overall slowdown in performance, and in fact, as the number of cores increases, we will see performance deteriorate below that obtainable on a single core. Luckily, we can deal with contention quite easily by adding a *contention manager* into the code (Line 21 in figures 1 and 2).

The most popular type of contention manager is exponential backoff: every time a CAS fails in tryPush() or tryPop(), the thread delays for a certain random time before attempting the CAS again. A thread will double the range from which it picks the random delay upon CAS failure, and will cut it in half upon CAS success. The randomized nature of the backoff scheme makes the timing of the thread's attempts to acquire the lock less dependent on the scheduler, reducing the chance of threads falling into a repetitive pattern in which they all try to CAS at the same time and end up starving. Contention managers[1,12,19] are key tools in the design of multicore data structures, even when no locks are used, and I expect them to play an even greater role as the number of cores grows.

Figure 3. The lock-free tryPush() and tryPop() methods.

```
1     public  class   LockFreeStack<T>  {
2       private  AtomicReference<Node> top =
3                    new AtomicReference<Node>(null);
4     ...
5
6       protected boolean tryPush(Node node) {
7         Node oldTop = top.get();
8         node.next = oldTop;
9        return top.compareAndSet(oldTop, node);
10      }
11
12      protected Node tryPop() throws EmptyException  {
13        Node oldTop = top.get();
14        if (oldTop == null) {
15           throw new EmptyException();
16        }
17        Node newTop = oldTop.next;
18        if  (top.compareAndSet(oldTop, newTop)) {
19            return oldTop;
20        } else {
21            return null ;
22        }
23      }
```

Figure 4. The EliminationBackoffStack<T>.



Each thread selects a random location in the array. If thread *A*'s pop() and thread *B*'s push() calls arrive at the same location at about the same time, then they exchange values without accessing the shared lock-free stack. A thread *C*, that does not meet another thread, eventually pops the shared lock-free stack.

A Lock-Free Stack

As noted, a drawback of our lock-based implementation, and in fact, of lock-based algorithms in general, is that the scheduler must guarantee that threads are preempted infrequently (or not at all) while holding the locks. Otherwise, other threads accessing the same locks will be delayed, and performance will suffer. This dependency on the capriciousness of the scheduler is particularly problematic in hard real-time systems where one requires a guarantee on how long method calls will take to complete.

We can eliminate this dependency by designing a lock-free stack implementation.[23] In the `LockFreeStack<T>`, instead of acquiring a lock to manipulate the stack, threads agree who can modify it by directly applying a CAS to the `top` variable. To do so, we only need to modify the code for the `tryPush()` and `tryPop()` methods, as in Figure 3. As before, if unsuccessful, the method calls are repeated after backing off, just as in the lock-based algorithm.

A quick analysis shows the completion of a `push` (respectively `pop`) method call cannot be delayed by the preemption of some thread: the stack's state is changed by a single CAS operation that either completes or not, leaving the stack ready for the next operation. Thus, a thread can only be delayed by scheduling infinitely many calls that successfully modify the top of the stack and cause the `tryPush()` to continuously fail. In other words, the system as a whole will always make progress no matter what the scheduler does. We call this form of progress *lock-freedom*. In many data structures, having at least some of the structure's methods be lock-free tends to improve overall performance.

It is easy to see that the lock-free stack is linearizable: it behaves like a sequential stack whose methods "take effect" at the points in time where their respective CAS on the top variable succeeded (or threw the exception in case of a `pop` on an empty stack). We can thus compose this stack with other linearizable objects without worrying about the implementation details: as far as safety goes, there is no difference between the lock-based and lock-free stacks.

An Elimination Backoff Stack

Like the lock-based stack, the lock-free stack implementation scales poorly, primarily because its single point of access forms a *sequential bottleneck*: method calls can proceed only one after the other, ordered by successful CAS calls applied to the stack's lock or top fields. A sad fact we should acknowledge is this sequential bottleneck is inherent: in the worst case it takes a thread at least $\Omega(n)$ steps and/or stalls (recall, a stall is the delay a thread incurs when it must wait for another thread taking a step) to push or pop a linearizable lock-free stack.[9] In other words, the theory tells us there is no way to avoid this bottleneck by distributing the stack implementation over multiple locations; there will always be an execution of linear complexity.

I expect the greatest change we will see is that concurrent data structures will go through a substantiative "relaxation process."

Surprisingly, though, we can introduce parallelism into many of the common case executions of a stack implementation. We do so by exploiting the following simple observation: if a `push` call is immediately followed by a `pop` call, the stack's state does not change; the two calls *eliminate* each other and it is as if both operations never happened. By causing concurrent pushes and pops to meet and pair up in separate memory locations, the thread calling `push` can exchange its value with a thread calling `pop`, without ever having to access the shared lock-free stack.

As depicted in Figure 4, in the `EliminationBackoffStack<T>`[11] one achieves this effect by adding an `EliminationArray` to the lock-free stack implementation. Each location in the array is a coordination structure called an *exchanger*,[16,18] an object that allows a pair of threads to rendezvous and exchange values.

Threads pick random array entries and try to pairup with complementary operations. The calls exchange values in the location in which they met, and return. A thread whose call cannot be eliminated, either because it has failed to find a partner, or because it found a partner with the wrong type of method call (such as a `push` meeting a `push`), can either try again to eliminate at a new location, or can access the shared lock-free stack. The combined data structure, array and stack, is linearizable because the lock-free stack is linearizable, and we can think of the eliminated calls as if they occurred at the point in which they exchanged values.

It is lock-free because we can easily implement a lock-free exchanger using a CAS operation, and the shared stack itself is already lock-free.

In the `EliminationBackoffStack`, the `EliminationArray` is used as a backoff scheme to a shared lock-free stack. Each thread first accesses the stack, and if it fails to complete its call (that is, the CAS attempt on top fails) because there is contention, it attempts to eliminate using the array instead of simply backing off in time. If it fails to eliminate, it calls the lockfree stack again, and so on. A thread dynamically selects the subrange of the array within which it tries to eliminate, growing and shrinking it exponentially in response to the load. Picking a smaller subrange allows a greater chance of a successful rendezvous when there are few threads, while a larger range lowers the chances of threads waiting on a busy `Exchanger` when the load is high.

In the worst case a thread can still fail on both the stack and the elimination. However, if contention is low, threads will quickly succeed in accessing the stack, and as it grows, there will be a higher number of successful eliminations, allowing many operations to complete in parallel in only a constant number of steps. Moreover, contention at the lock-free stack is reduced because eliminated operations never access the stack. Note that we described a lock-free implementation, but, as with many concurrent data structures, on some systems a lock-based implementation might be more fitting and deliver better performance.

An Elimination Tree

A drawback of the elimination backoff stack is that under very high loads the number of un-eliminated threads accessing the shared lock-free stack may remain high, and these threads will continue to have linear complexity. Moreover, if we have, say, bursts of `push` calls followed by bursts of `pop` calls, there will again be no elimination and therefore no parallelism. The problem seems to be our insistence on having a linearizable stack: we devised a distributed solution that cuts down on the number of stalls, but the theoretical worst case linear time scenario can happen too often.

This leads us to try an alternative approach: relaxing the consistency condition for the stack. Instead of a linearizable stack, let's implement a quiescently consistent one.[4,14] A stack is quiescently consistent if in any execution, whenever there are no ongoing `push` and `pop` calls, it meets the LIFO stack specification for all the calls that preceded it. In other words, quiescent consistency is like a game of musical chairs, we map the object to the sequential specification when and only when the music stops. As we will see, this relaxation will nevertheless provide quite powerful semantics for the data structure. In particular, as with linearizability, quiescent consistency allows objects to be composed as black boxes without having to know anything about their actual implementation.

Consider a binary tree of objects called *balancers* with a single input wire and two output wires, as depicted in Figure 5. As threads arrive at a balancer, it repeatedly sends them to the top wire and then the bottom one, so its top wire always has one more thread than the bottom wire. The `Tree[k]` network is a binary tree of balancers constructed inductively by placing a balancer before two `Tree[k/2]` networks of balancers and not shuffling their outputs.[22]

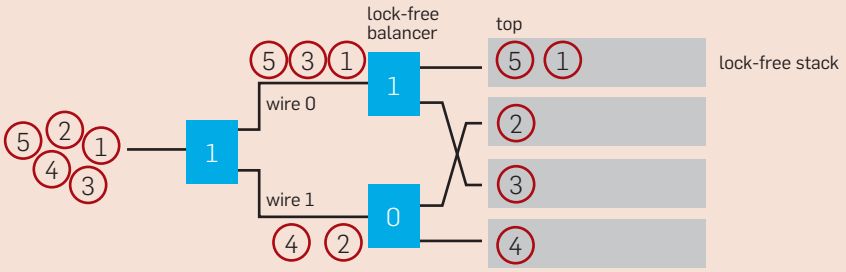
We add a collection of lock-free stacks to the output wires of the tree. To perform a `push`, threads traverse the balancers from the root to the leaves and then `push` the item onto the appropriate stack. In any quiescent state, when there are no threads in the tree, the output items are balanced out so that the top stacks have at most one more than the bottom ones, and there are no gaps.

We can implement the balancers in a straightforward way using a bit that threads toggle: they fetch the bit and then complement it (a CAS operation), exiting on the output wire they fetched (zero or one). How do we perform a pop? Magically, to perform a pop threads traverse the balancers in the opposite order of the push, that is, in each balancer, after complementing the bit, they follow this complement, the opposite of the bit they fetched. Try this; you will see that from one quiescent state to the next, the items removed are the last ones pushed onto the stack. We thus have a collection of stacks that are accessed in parallel, yet act as one quiescent LIFO stack.

The bad news is that our implementation of the balancers using a bit means that every thread that enters the tree accesses the same bit in the root balancer, causing that balancer to become a bottleneck. This is true, though to a lesser extent, with balancers lower in the tree.

We can parallelize the tree by exploiting a simple observation similar to one we made about the elimination backoff stack:

Figure 5. A `Tree[4]` network leading to four lock-free stacks.



Threads pushing items arrive at the balancers in the order of their numbers, eventually pushing items onto the stacks located on their output wires. In each balancer, a pushing thread fetches and then complements the bit, following the wire indicated by the fetched value (If the state is 0 the pushing thread it will change it to 1 and continue to wire 0, and if it was 1 will change it to 0 and continue on wire 1). The tree and stacks will end up in the balanced state seen in the figure. The state of the bits corresponds to 5 being the last item, and the next location a pushed item will end up on is the lock-free stack containing item 2. Try it! A popping thread does the opposite of the pushing one: it complements the bit and follows the complemented value. Thus, if a thread executes a pop in the depicted state, it will end up switching a 1 to a 0 at the top balancer, and leave on wire 0, then reach the top 2nd level balancer, again switching a 1 to a 0 and following its 0 wire, ending up popping the last value 5 as desired. This behavior will be true for concurrent executions as well: the sequences of values in the stacks in all quiescent states can be shown to preserve LIFO order.

If an *even* number of threads passes through a balancer, the outputs are evenly balanced on the top and bottom wires, but the balancer's state remains unchanged.

The idea behind the `EliminationTree<T>`[20,22] is to place an `EliminationArray` in front of the bit in every balancer as in Figure 6. If two popping threads meet in the array, they leave on opposite wires, without a need to touch the bit, as anyhow it would have remained in its original state. If two pushing threads meet in the array, they also leave on opposite wires. If a push or pop call does not manage to meet another in the array, it toggles the bit and leaves accordingly. Finally, if a push and a pop meet, they eliminate, exchanging items as in the `EliminationBackoffStack`. It can be shown that this implementation provides a quiescently consistent stack,[a] in which, in most cases, it takes a thread $O(\log k)$ steps to complete a push or a pop, where k is the number of lock-free stacks on its output wires.

A Pool Made of Stacks

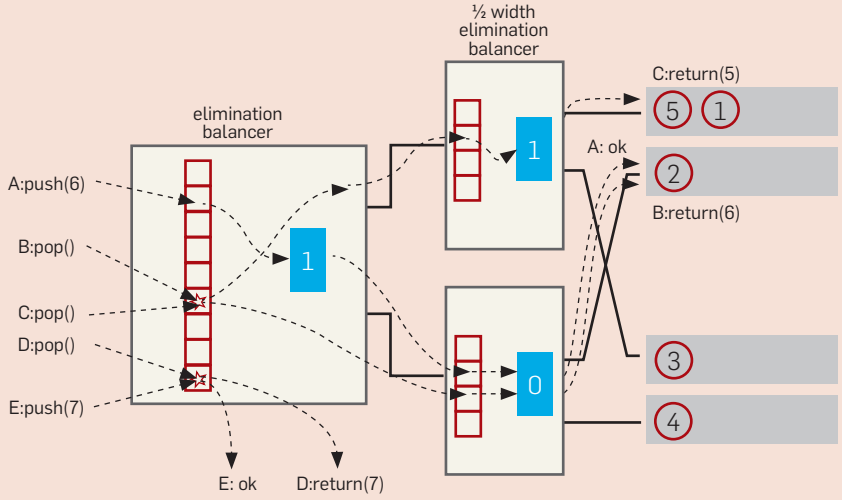
The collection of stacks accessed in parallel in the elimination tree provides quiescently consistent LIFO ordering with a high degree of parallelism. However, each method call involves a logarithmic number of memory accesses, each involving a CAS operation, and these accesses are not localized, that is, threads are repeatedly accessing locations they did not access recently.

This brings us to the final two issues one must take into account when designing concurrent data structures: the machine's memory hierarchy and its coherence mechanisms. Mainstream multicore architectures are cache coherent, where on most machines the L2 cache (and in the near future the L3 cache as well) is shared by all cores. A large part of the machine's performance on shared data is derived from the threads' ability to find the data cached. The shared caches are unfortunately a bounded resource, both in their size and in the level of access parallelism they offer. Thus, the data structure design needs to attempt to lower the overall number of accesses to memory, and to maintain locality as much as possible.

What are the implications for our stack design? Consider completely relaxing the LIFO property in favor of a `Pool<T>` structure in which there is no temporal ordering on `push()` and `pop()` calls. We will provide a concurrent lock-free implementation of a pool that supports high parallelism, high locality, and has a low cost in terms of the overall number of accesses to memory. How useful is such a concurrent pool? I would like to believe that most concurrent applications can be tailored to use pools in place of queues and stacks (perhaps with some added liveness conditions)...time will tell.

Our overall concurrent pool design is quite simple. As depicted in Figure 7, we allocate a collection of n concurrent lock-free stacks, one per computing thread (alternately we could allocate one stack per collection of threads on the same core, depending on the specific machine architecture). Each thread will push and pop from its own assigned stack. If, when it attempts to pop, it finds its own stack is empty, it will repeatedly attempt to "steal" an item from another randomly chosen stack.[b] The pool has, in

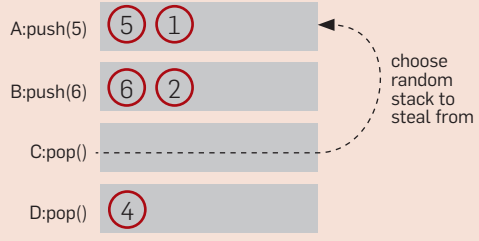
Figure 6. The `EliminationTree<T>`.



Each balancer in `Tree[4]` is an elimination balancer. The state depicted is the same as in Figure 5. From this state, a push of item 6 by thread A will not meet any others on the elimination arrays and so will toggle the bits and end up on the 2nd stack from the top. Two pops by threads B and C will meet in the top balancer's array and end up going up and down without touching the bit, ending up popping the last two values 5 and 6 from the top two lock-free stacks. Finally, threads D and E will meet in the top array and "eliminate" each other, exchanging the value 7 and leaving the tree. This does not ruin the tree's state since the states of all the balancers would have been the same even if the threads had both traversed all the way down without meeting: they would have anyhow followed the same path down and ended up exchanging values via the same stack.

Figure 7. The concurrent `Pool<T>`.

Each thread performs `push()` and `pop()` calls on a lock-free stack and attempts to steal from other stacks when a `pop()` finds the local stack empty. In the figure, thread *C* will randomly select the top lock-free stack, stealing the value 5. If the lock-free stacks are replaced by lock-free deques, thread *C* will pop the oldest value, returning 1.



a To keep things simple, pop operations should block until a matching push appears.

the common case, the same $O(1)$ complexity per method call as the original lockfree stack, yet provides a very high degree of parallelism. The act of stealing itself may be expensive, especially when the pool is almost empty, but there are various techniques to reduce the number of steal attempts if they are unlikely to succeed. The randomization serves the purpose of guaranteeing an even distribution of threads over the stacks, so that if there are items to be popped, they will be found quickly. Thus, our construction has relaxed the specification by removing the causal ordering on method calls and replacing the deterministic liveness and complexity guarantees with probabilistic ones.

As the reader can imagine, the $O(1)$ step complexity does not tell the whole story. Threads accessing the pool will tend to pop items that they themselves recently pushed onto their own designated stack, therefore exhibiting good cache locality. Moreover, since chances of a concurrent stealer are low, most of the time a thread accesses its lock-free stack alone. This observation allows designers to create a lockfree "stack-like" structure called a `Deque`[c] that allows the frequently accessing local thread to use only loads and stores in its methods, resorting to more expensive CAS based method calls only when chances of synchronization with a conflicting stealing thread are high.[3,6]

The end result is a pool implementation that is tailored to the costs of the machine's memory hierarchy and synchronization operations. The big hope is that as we go forward, many of these architecture-conscious optimizations, which can greatly influence performance, will move into the realm of compilers and concurrency libraries, and the need for everyday programmers to be aware of them will diminish.

What Next?

The pool structure ended our sequence of relaxations. I hope the reader has come to realize how strongly the choice of structure depends on the machine's size and the application's concurrency requirements. For example, small collections of threads can effectively share a lock-based or lock-free stack, slightly larger ones an elimination stack, but for hundreds of threads we will have to bite the bullet and move from a stack to a pool (though within the pool implementation threads residing on the same core or machine cluster could use a single stack quite effectively).

In the end, we gave up the stack's LIFO ordering in the name of performance. I imagine we will have to do the same for other data structure classes. For example, I would guess that search structures will move away from being comparison based, allowing us to use hashing and similar naturally parallel techniques, and that priority queues will have a relaxed priority ordering in place of the strong one imposed by deleting the minimum key. I can't wait to see what these and other structures will look like.

As we go forward, we will also need to take into account the evolution of hardware support for synchronization. Today's primary construct, the CAS operation, works on a single memory location. Future architectures will most likely support synchronization techniques such as transactional memory,[13,21] allowing threads to instantaneously read and write multiple locations in one indivisible step. Perhaps more important than the introduction of new features like transactional memory is the fact that the relative costs of synchronization and coherence are likely to change dramatically as new generations of multicore chips role out. We will have to make sure to consider this evolution path carefully as we set our language and software development goals.

Concurrent data structure design has, for many years, been moving forward at glacial pace. Multicore processors are about to heat things up, leaving us, the data structure designers and users, with the interesting job of directing which way they flow. Let's try to get it right. C

c This `Deque` supports `push()` and `pop()` methods with the traditional LIFO semantics and an additional `popTop()` method for stealers that pops the first-in (oldest) item.[5]

References

1. Agarwal, A. and Cherian, M. Adaptive backoff synchronization techniques. In *Proceedings of the 16th International Symposium on Computer Architecture* (May 1989), 396–406.
2. Anderson, J. and Kim, Y. An improved lower bound for the time complexity of mutual exclusion. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing* (2001), 90–99.
3. Arora, N.S., Blumofe, R.D. and Plaxton, C.G. Thread scheduling for multiprogrammed multiprocessors. *Theory of Computing Systems 34*, 2 (2001), 115–144.
4. Aspnes, J., Herlihy, M. and Shavit, N. Counting networks. *J. ACM 41*, 5 (1994), 1020–1048.
5. Blumofe, R.D. and Leiserson, C.E. Scheduling multithreaded computations by work stealing. *J. ACM 46*, 5 (1999), 720–748.
6. Chase, D. and Lev, Y. Dynamic circular work-stealing deque. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures* (2005). ACM Press, NY, 21–28.
7. Cypher, R. The communication requirements of mutual exclusion. In *ACM Proceedings of the Seventh Annual Symposium on Parallel Algorithms and Architectures* (1995), 147-156.
8. Dwork, C., Herlihy, M. and Waarts, O. Contention in shared memory algorithms. *J. ACM 44*, 6 (1997), 779–805.
9. Fich, F.E., Hendler, D. and Shavit, N. Linear lower bounds on real-world implementations of concurrent objects. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (2005).IEEE Computer Society, Washington, D.C., 165–173.
10. Gibbons, P.B., Matias, Y. and Ramachandran, V. The queue-read queue-write PRAM model: Accounting for contention in parallel algorithms. *SIAM J. Computing 28*, 2 (1999), 733–769.
11. Hendler, D., Shavit, N. and Yerushalmi, L. A scalable lock-free stack algorithm. *J. Parallel and Distributed Computing 70*, 1 (Jan. 2010), 1–12.
12. Herlihy, M., Luchangco, V., Moir, M. and Scherer III, W.N. Software transactional memory for dynamic-sized data structures. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*. ACM, NY, 2003, 92–101.
13. Herlihy, M. and Moss, E. Transactional memory: architectural support for lock-free data structures. *SIGARCH Comput. Archit. News 21*, 2 (1993), 289–300.
14. Herlihy, M. and Shavit, N. *The Art of Multiprocessor Programming*. Morgan Kaufmann, San Mateo, CA, 2008.
15. Herlihy, M. and Wing, J. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Programming Languages and Systems 12*, 3 (July 1990), 463–492.
16. Moir, M., Nussbaum, D., Shalev, O. and Shavit, N. Using elimination to implement scalable and lock-free fifo queues. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures*. ACM Press, NY, 2005, 253–262.
17. Moir, M. and Shavit, N. Concurrent data structures. *Handbook of Data Structures and Applications*, D. Metha and S. Sahni, eds. Chapman and Hall/CRC Press, 2007, 47-14, 47-30.
18. Scherer III, W.N., Lea, D. and Scott, M.L. Scalable synchronous queues. In *Proceedings of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM Press, NY, 2006, 147–156.
19. Scherer III, W.N. and Scott, M.L. Advanced contention management for dynamic software transactional memory. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing*. ACM, NY, 2005, 240–248.
20. Shavit, N. and Touitou, D. Elimination trees and the construction of pools and stacks. Theory of Computing Systems 30 (1997), 645–670.
21. Shavit, N. and Touitou, D. Software transactional memory. *Distributed Computing 10*, 2 (Feb. 1997), 99–116.
22. Shavit, N. and Zemach, A. Diffracting trees. *ACM Transactions on Computer Systems 14*, 4 (1996), 385–428.
23. Treiber, R.K. Systems programming: Coping with parallelism. Technical Report RJ 5118 (Apr. 1986). IBM Almaden Research Center, San Jose, CA.

Nir Shavit is a professor of computer science at Tel-Aviv University and a member of the Scalable Synchronization Group at Oracle Labs. He is a recipient of the 2004 ACM/EATCS Gödel Prize.

research highlights

DOI:10.1145/1897852.1897874

Technical Perspective
Concerto for Violin and Markov Model

By Juan Bello, Yann LeCun, and Robert Rowe

IN THE OPENING moments of Jean Sibelius' Violin Concerto, the young soloist plays delicately, almost languidly. The orchestra responds in kind, muting the repeated string motif to a whisper. As the piece progresses, soloist and orchestra alternatively perform the main motifs in increasing measures of power and virtuosity, which inexorably lead toward the movement's stirring resolution. The soloist looks relieved as she crosses the stage to shake the conductor's hand.

This violinist, like most others in music education, can benefit enormously from interacting with large ensembles in honing her performing skills. However, the demand far exceeds the number and capabilities of existing orchestras, ensuring most of these students won't have access to this experience. Our soloist is no exception. The previous paragraph describes her interaction with Chris Raphael's Music Plus One system: A machine learning-driven alternative to working with orchestras that retains much of the expressivity and interactivity that makes concerto performance such a rewarding and educational experience. The following paper details the approach, for videos see http://www.music.informatics.indiana.edu/papers/icml10/.

Automatic music accompaniment has been actively researched since the 1980s, starting with the work of such pioneers as Barry Vercoe and Roger Dannenberg. The problem can be broken into three subparts:[1] tracking the playing of a human soloist, matching it to a known musical score, and synthesizing an appropriate accompaniment to that solo part in real time. Solutions usually involve ingenious pattern-matching mechanisms for dealing with expressive, incorrectly played or missing notes in the soloist performance, while using the output of the pattern match to drive the scheduling of accompaniment events. However, as Raphael notes, it is impossible to accomplish score following by reaction alone. The system must incorporate a predictive component that attempts to align upcoming notes of the accompaniment with imminent attacks of the human player. Failing to solve this problem can result in potentially disastrous consequences for the performance.

The proposed approach starts by using a hidden Markov model-based score follower tasked with estimating the start time of the notes played by the soloist and matching them to a position in the score. The model considers the sequence of frame-wise signal features, characterizing transient and pitch information on the audio input, as its output, and the state graph for the Markov chain as a sequence of note sub-graphs modeling the soloist's performance. In a process akin to echo cancellation, the contribution of the accompanist to the audio signal is explicitly modeled to avoid the system following itself.

The estimated sequence of note onset times could be used to adaptively control the playback rate of the orchestral recording and match the player's position in the score. However, it is not possible to accurately estimate the soloist's timing without a certain amount of latency, thus causing the orchestra to consistently lag behind. The author's solution is to use a Gaussian graphical model to predict the timing of the next orchestra event based on previous observation of both solo and orchestra note occurrences. In this context, the orchestra's playback rate, modulated using the well-established phase vocoder method, is continuously re-estimated as new events are observed, a formulation that is robust to missing notes, as pending orchestra notes are conditioned only on those events that have been observed. Crucially, Raphael uses information from rehearsals to adapt the predictive model to the soloist's interpretation style, thus mimicking the process by which human performers learn to anticipate each other's actions through practice.

The system's architecture is constructed using common staples of the machine learning literature such as hidden Markov models and Gaussian graphical models. Yet, these elements and others are combined using a healthy dose of musically meaningful insights, as well as the engineering acumen necessary to make the system work robustly in real time, as emphatically demonstrated by the accompanying videos. The result is an effective, albeit limited, model of a human accompanist that has been extensively tested by student performers at one of the country's premier conservatories, the Jacobs School of Music at the University of Indiana.

The system is an important milestone toward *machine musicianship*. Through the use of machine learning, computers are acquiring new skills once thought to be uniquely human. If music communicates human emotions, isn't it futile to teach computers to play music? The beauty of Music Plus One is that it follows and amplifies the emotions of the human player. In that sense, it is very much like a traditional musical instrument, albeit a highly sophisticated one. Music Plus One relies on a predetermined musical score. Perhaps the next step would be to create an automatic rhythm section that reacts to a soloist the way jazz musicians instantly react to each other's improvisations. It would require a new level of machine musicianship, and would constitute a major challenge for machine learning, one that will increase our understanding and appreciation of the human mind's ability to create and improvise. C

Reference

1. Dannenberg, R.B. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference* (Paris, France, 1984), 193–198.

Juan Bello is an assistant professor of music technology in the Department of Music and Performing Arts Professions at NYU's Steinhardt School of Culture, Education and Human Development.

Yann LeCun is Silver Professor of Computer Science and Neural Science at NYU and co-founder of MuseAmi, a music technology company.

Robert Rowe is a professor and vice chair in the Department of Music and Performing Arts Professions at NYU's Steinhardt School of Culture, Education and Human Development, where he directs the Music Composition Program.

DOI:10.1145/1897852.1897875

The Informatics Philharmonic

By Christopher Raphael

Abstract

A system for musical accompaniment is presented in which a computer-driven orchestra follows and learns from a soloist in a concerto-like setting. The system is decomposed into three modules: The first computes a real-time score match using a hidden Markov model; the second generates the output audio by phase-vocoding a preexisting audio recording; the third provides a link between these two, by predicting future timing evolution using a Kalman Filter–like model. Several examples are presented showing the system in action in diverse musical settings. Connections with machine learning are highlighted, showing current weaknesses and new possible directions.

1. MUSICAL ACCOMPANIMENT SYSTEMS

Musical accompaniment systems are computer programs that serve as musical partners for live musicians, usually playing a supporting role for music centering around the live player. The types of possible interaction between live player and computer are widely varied. Some approaches create sound by processing the musician's audio, often driven by analysis of the audio content itself, perhaps distorting, echoing, harmonizing, or commenting on the soloist's audio in largely predefined ways.[8, 12] Other orientations are directed toward improvisatory music, such as jazz, in which the computer follows the outline of a score, perhaps even composing its own musical part "on the fly,"[3] or evolving as a "call and response" in which the computer and human alternate the lead role.[6, 9] Our focus here is on a third approach that models the traditional "classical" concerto-type setting in which the computer performs a precomposed musical part in a way that follows a live soloist.[2, 4, 11] This categorization is only meant to summarize some past work, while acknowledging that there is considerable room for blending these scenarios, or working entirely outside this realm of possibilities.

The motivation for the *concerto* version of the problem is strikingly evident in the Jacobs School of Music (JSoM) at Indiana University, where most of our recent experiments have been performed. For example, the JSoM contains about 200 student pianists, for whom the concerto literature is central to their daily practice and aspirations. However, in the JSoM, the regular orchestras perform only two piano concerti each year using student soloists, thus ensuring that most of these aspiring pianists will never perform as orchestral soloist while at IU. We believe this is truly unfortunate since nearly all of these students have the necessary technical skills and musical depth to greatly benefit from the concerto experience. Our work in musical accompaniment systems strives to bring this rewarding experience to the music students, amateurs, and many others who would like to play as orchestral soloist, though, for whatever reason, do not have the opportunity.

Even within the realm of classical music, there are a number of ways to further subdivide the accompaniment problem, requiring substantially different approaches. The JSoM is home to a large string pedagogy program beginning with students at 5 years of age. Students in this program play solo pieces with piano even in their first year. When accompanying these early-stage musicians, the pianist's role is not simply to *follow* the young soloist, but to *teach* as well, by modeling good rhythm, steady tempo where appropriate, while introducing musical ideas. In a sense, this is the hardest of all classical music accompaniment problems, since the accompanist must be expected to know *more* than the soloist, thus dictating when the accompanist should follow, as well as when and *how* to lead. A coarse approximation to this accompanist role provides a rather rigid accompaniment that is not overly responsive to the soloist's interpretation (or errors)—there are several commercial programs that take this approach. The more sophisticated view of the pedagogical music system—one that follows and leads as appropriate—is almost completely untouched, possibly due to the difficulty of modeling the objectives. However, we see this area as fertile for lasting research contributions and hope that we, and others, will be able to contribute to this cause.

An entirely different scenario deals with music that evolves largely without any traditional sense of rhythmic flow, such as in some compositions of Penderecki, Xenakis, Boulez, Cage, and Stockhausen, to name some of the more famous examples. Such music is often notated in terms of seconds, rather than beats or measures, to emphasize the irrelevance of regular pulse. For works of this type involving soloist and accompaniment, the score can indicate points of synchronicity, or time relations, between various points in the solo and accompaniment parts. If the approach is based solely on audio, a natural strategy is simply to wait until various solo events are detected, and then to *respond* to these events. This is the approach taken by the IRCAM score follower, with some success in a variety of pieces of this type.[2]

A third scenario, which includes our system, treats works for soloist and accompaniment having a continuing musical pulse, including the overwhelming majority of "common practice" art music. This music is the primary focus of most of our performance-oriented music students at the JSoM, and is the music where our accompaniment system is most at home. Music containing a regular, though not rigid, pulse requires close synchronization between the solo and accompanying parts, as the overall result suffers greatly as this

The original version of this chapter is entitled "Music Plus One and Machine Learning" and was published in *Proceedings of the International Conference on Machine Learning*, Haifa, 2010.

synchrony degrades.

Our system is known interchangeably as the "Informatics Philharmonic," or "Music Plus One" (MPO), due to its alleged improvement on the play-along accompaniment records from the *Music Minus One* company that inspired our work. For several years, we have collaborated with faculty and students in the JSoM on this traditional concerto setting, in an ongoing effort to improve the performance of our system while exploring variations on this scenario. The web page http://www.music.informatics.indiana.edu/papers/icml10 contains a video of violinist Yoo-jin Cho, accompanied by our system on the first movement of the Sibelius violin concerto, taken from a lecture/concert for our *Art's Week* festival of 2007. We will present a description of the overall architecture of our system in terms of its three basic components: Listen, Predict, and Play, including several illuminating examples. We also identify open problems or limitations of proposed approaches that are likely to be interesting to the Machine Learning community, and well may benefit from their contributions.

The basic technology required for common practice classical music extends naturally to the *avant garde* domain. In fact, we believe one of the greatest potential contributions of the accompaniment system is in new music composed specifically for human–computer partnerships. The computer offers essentially unlimited virtuosity in terms of playing fast notes and coordinating complicated rhythms. On the other hand, at present, the computer is comparatively weak at providing aesthetically satisfying musical interpretations. Compositions that leverage the technical ability of the accompaniment system, while humanizing the performance through the live soloist's leadership, provide an open-ended musical meeting place for the twenty-first-century composition and technology. Several compositions of this variety, written specifically for our accompaniment system by Swiss composer and mathematician Jan Beran, are presented at the web page referenced above.

2. OVERVIEW OF MUSIC PLUS ONE

Our system is composed of three sub-tasks called "Listen," "Predict," and "Play." The Listen module interprets the audio input of the live soloist as it accumulates in real time. In essence, Listen annotates the incoming audio with a "running commentary," identifying note onsets with variable detection latency, using the hidden Markov model discussed in Section 3. A moment's thought here reveals that some detection latency is inevitable since a note must be heard for an instant before it can be identified. For this reason, we believe it is hopeless to build a purely "responsive" system—one that waits until a solo note is detected before playing a synchronous accompaniment event: Our detection latency is usually in the 30–90-ms range, enough to prove fatal if the accompaniment is consistently behind by this much. For this reason, we model the timing of our accompaniment on the human musician, continually predicting future evolution, while modifying these predictions as more information becomes available. The module of our system that performs this task, Predict, is a Gaussian graphical model quite close to a Kalman Filter, discussed in Section 4. The Play module uses phase-vocoding[5] to construct the orchestral audio output using audio from an accompaniment-only recording. This well-known technique warps the timing of the original audio without introducing pitch distortions, thus retaining much of the original musical intent including balance, expression, and tone color. The Play process is driven by the output of the Predict module, in essence by following an evolving sequence of future targets like a trail of breadcrumbs.

While the basic methodology of the system relies on old standards from the ML community—HMMs and Gaussian graphical models—the computational challenge of the system should not be underestimated, requiring accurate real-time two-way audio computation in musical scenarios complex enough to be of interest in a sophisticated musical community. The system was implemented for off-the-shelf hardware in C and C++ over a period of more than 15 years by the author. Both Listen and Play are implemented as separate threads which both make calls to the Predict module when either a solo note is detected (Listen) or an orchestra note is played (Play).

What follows is a more detailed look at Listen and Predict.

3. LISTEN: HMM-BASED SCORE FOLLOWING

Blind music audio recognition[1, 7, 13] treats the automatic transcription of music audio into symbolic music representations, using no prior knowledge of the music to be recognized. This problem remains completely open, especially with polyphonic (several independent parts) music, where the state of the art remains primitive. While there are many ways one can build reasonable data models quantifying how well a particular audio instant matches a hypothesized collection of pitches, what seems to be missing is the musical language model. If phonemes and notes are regarded as the atoms of speech and music, there does not seem to be a musical equivalent of the *word*. Furthermore, while music follows simple logic and can be quite predictable, this logic is often cast in terms of higher-level constructs such as meter, harmony, and motivic transformation. Computationally tractable models such as note *n*-grams seem to contribute very little here, while a computationally useful music language model remains uncharted territory.

Our Listen module deals with the much simpler situation in which the music *score* is known, giving the pitches the soloist will play along with their approximate durations. Thus, the score following problem is one of *alignment* rather than *recognition*. Score following, otherwise known as online alignment, is more difficult than its offline cousin, since an online algorithm cannot consider future audio data in estimating the times of audio events. A score following must "hear" a little bit of a note before the note's onset can be detected, thus always resulting with some degree of *latency*—the lag between the estimated onset time and the time the estimate is made. One of the principal challenges of online alignment is navigating the trade-off between latency and accuracy. Schwarz[14] gives a nice annotated bibliography of the many contributions to score following.

3.1. The listen model

Our HMM approach views the audio data as a sequence of "frames," $y_1, y_2, \ldots, y_T$, with about 30 frames per second, while modeling these frames as the output of a hidden Markov chain, $x_1, x_2, \ldots, x_T$. The state graph for the Markov chain, described in Figure 1, models the music as a sequence of sub-graphs, one for each solo note, arranged so that the process enters the start of the $(n+1)$th note as it leaves the nth note. From the figure, one can see that each note begins with a short sequence of states meant to capture the *attack* portion of the note. This is followed by another sequence of states with self-loops meant to capture the main body of the note, and to account for the variation in note duration we may observe, as follows.

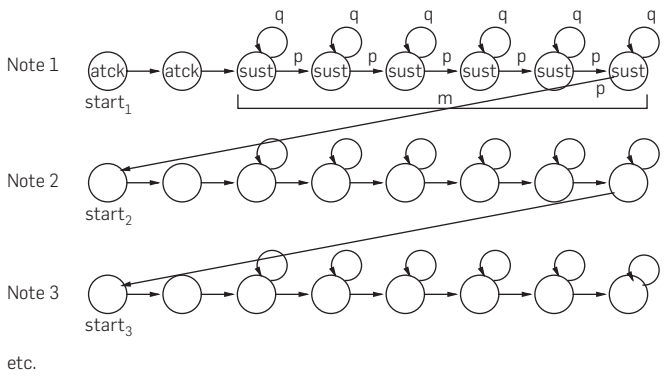
If we chain together m states which each either move forward, with probability p, or remain in the current state, with probability $q = 1 - p$, then the total number of state visits (audio frames), L, spent in the sequence of m states has a negative binomial distribution

$$P(L=l)=\binom{m-1}{l-1}p^m q^{l-m}$$

for $l = m, m+1, \ldots$. While convenient to represent this distribution with a Markov chain, the asymmetric nature of the negative binomial is also musically reasonable: While it is common for an inter-onset interval (IOI) to be much longer than its nominal length, the reverse is much less common. For each note, we choose the parameters m and p so that $E(T) = m/p$ and $\mathrm{Var}(T) = mq/p^2$ reflect our prior beliefs. Before any rehearsals, the mean is chosen to be consistent with the note value and the nominal tempo given in the score, while the variance is chosen to be a fixed increasing function of the mean. However, once we have rehearsed a piece a few times, we choose m and p according to the method of moments—so that the empirical mean and variance agree with the mean and variance from the model.

In reality, we use a wider variety of note models than depicted in Figure 1, with variants for short notes, notes ending with optional rests, notes that are rests, etc., though all follow the same essential idea. The result is a network of thousands of states.

Figure 1. The state graph for the hidden sequence, $x_1, x_2, \ldots$, of our HMM.



Our data model is composed of three features $b_t(y_t)$, $e_t(y_t)$, $s_t(y_t)$ assumed to be conditionally independent given the state:

$$P(b_t, e_t, s_t | x_t) = P(b_t | x_t)\, P(e_t | x_t)\, P(s_t | x_t).$$

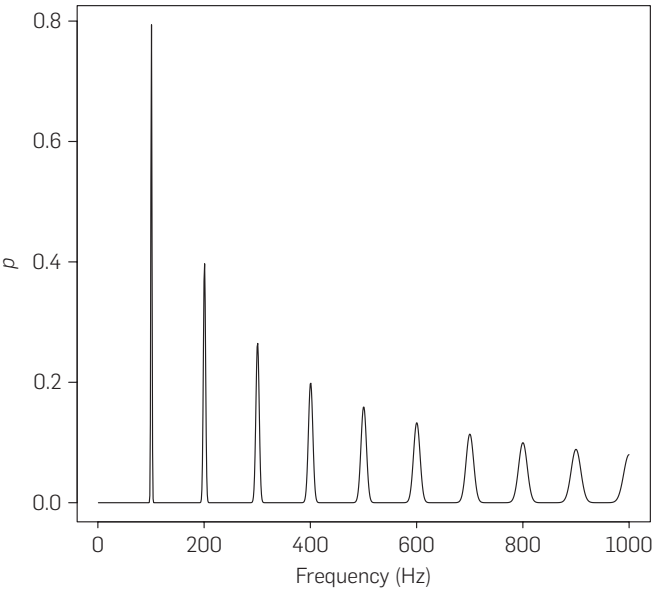
The first feature, b_t, measures the local "burstiness" of the signal, particularly useful in distinguishing between note attacks and steady-state behavior—observe that we distinguished between the attack portion of a note and steady-state portion in Figure 1. The second feature, e_t, measures the local energy, useful in distinguishing between rests and notes. By far, however, the vector-valued feature s_t is the most important, as it is well-suited to making pitch discriminations, as follows.

We let f_n denote the frequency associated with the nominal pitch of the nth score note. As with any quasi-periodic signal with frequency f_n, we expect that the audio data from the nth note will have a magnitude spectrum composed of "peaks" at integral multiples of f_n. This is modeled by the Gaussian mixture model depicted in Figure 2

$$p_n(j) = \sum_{h=1}^{H} w_h N(j;\, \mu = h f_n, \sigma^2 = (\rho h f_n)^2)$$

where $\Sigma_h w_h = 1$ and $N(j; \mu, \sigma^2)$ is a discrete approximation of a Gaussian distribution. The model captures the note's "spectral envelope," describing the way energy is distributed over the frequency range. In addition, due to the logarithmic nature of pitch, frequency "errors" committed by the player are proportional to the desired frequency. This is captured in our model by the increasing variance of the mixture components. We define s_t to be the magnitude spectrum of y_t, normalized to sum to constant value, C. If we believe the nth note is sounding in the tth frame, we regard

Figure 2. An idealized note spectrum modeled as a mixture of Gaussians.



s_t as the histogram of a random sample of size C. Thus our data model becomes the multinomial distribution

$$P(s_t \mid x_t \in \text{ note } n) = \frac{C!}{\prod_j s_t(j)!} \prod_j p_n(j)^{s_t(j)} \quad (1)$$

It is worth noting that the model generalizes in a straightforward way to situations in which multiple pitches sound at once, simply by mixing several distributions of the forms of Equation 3.1. In this way our approach accommodates anything from double stops on the violin to large ensemble performances.

This modeling approach describes the part of the audio spectrum due to the soloist reasonably well. However, our actual signal will receive not only this solo contribution, but also audio *generated* by our accompaniment system itself. If the accompaniment audio contains frequency content that is confused with the solo audio, the result is the highly undesirable possibility of the accompaniment system *following itself*—in essence, chasing its own shadow. To a certain degree, the likelihood of this outcome can be diminished by "turning off" the score follower when the soloist is not playing; of course we do this. However, there is still significant potential for shadow-chasing since the pitch content of the solo and accompaniment parts is often similar.

Our solution is to directly model the accompaniment contribution to the audio signal we receive. Since we *know* what the orchestra is playing (our system generates this audio), we add this contribution to the data model. More explicitly, if q_t is the magnitude spectrum of the orchestra's contribution in frame t, we model the conditional distribution of s_t using Equation 1, but with $p_{t,n} = \lambda p_n + (1 - \lambda) q_t$ for $0 < \lambda < 1$ instead of p_n.

This addition creates significantly better results in many situations. The surprising difficulty in actually implementing the approach, however, is that there seems to be only weak agreement between the *known* audio that our system plays through the speakers and the accompaniment audio that *comes back* through the microphone. Still, with various averaging tricks in the estimation of q_t, we can nearly eliminate the undesirable shadow-chasing behavior.

3.2. Online interpretation of audio

One of the worst things a score follower can do is report events before they have occurred. In addition to the sheer impossibility of producing accurate estimates in this case, the musical result often involves the accompanist arriving at a point of coincidence before the soloist does. When the accompanist "steps on" the soloist in this manner, the soloist must struggle to regain control of the performance, perhaps feeling desperate and irrelevant in the process. Since the consequences of false positives are so great, the score follower must be reasonably certain that a note event has already occurred before reporting its location. The probabilistic formulation of online score following is the key to avoiding such false positives, while navigating the accuracy-latency trade-off in a reasonable manner.

Every time we process a new frame of audio we recompute the "forward" probabilities, $p(x_t|y_1, \ldots, y_t)$, for our current frame, t. Listen waits to detect note n until we are sufficiently confident that its onset is in the past. That is, until

$$P(x_t \geq \text{start}_n | y_1, \ldots, y_t) \geq \tau$$

for some constant, τ. In this expression, start_n represents the initial state of the nth note model, as indicated in Figure 1, which is either before, or after all other states in the model ($x_t \geq \text{start}_n$ makes sense here). Suppose that t^* is the first frame where the above inequality holds. When this occurs, our knowledge of the note onset time can be summarized by the function of t:

$$P(x_t = \text{start}_n | y_1, \ldots, y_{t^*})$$

which we compute using the forward–backward algorithm. Occasionally this distribution conveys uncertainty about the onset time of the note, say, for instance, if it has high variance or is bimodal. In such a case we simply do not report the onset time of the particular note, believing it is better to remain silent than provide bad information. Otherwise, we estimate the onset as

$$\hat{t}_n = \arg\max_{t \leq t^*} P(x_t = \text{start}_n | y_1, \ldots, y_{t*}) \quad (2)$$

and deliver this information to the Predict module.

Several videos demonstrating the ability of our score following can be seen at the aforementioned web site. One of these simply plays the audio while highlighting the locations of note onset detections at the times they are made, thus demonstrating detection latency—what one *sees* lags slightly behind what one *hears*. A second video shows a rather eccentric performer who ornaments wildly, makes extreme tempo changes, plays wrong notes, and even repeats a measure, thus demonstrating the robustness of the score follower.

4. PREDICT: MODELING MUSICAL TIMING

As discussed in Section 2, we believe a purely *responsive* accompaniment system cannot achieve acceptable coordination of parts in the range of common practice "classical" music we treat, thus we choose to schedule our accompaniment through prediction rather than response. Our approach is based on a probabilistic model for musical timing. In developing this model, we begin with three important traits we believe such a model must have.

1. Since our accompaniment must be constructed in real time, the computational demand of our model must be feasible in real time.
2. Our system must improve with rehearsal. Thus our model must be able to automatically train its parameters to embody the timing nuances demonstrated by the live player in past examples. This way our system can better *anticipate* the future musical evolution of the current performance.
3. If our rehearsals are to be successful in guiding the system toward the desired musical end, the system must "sightread" (perform without rehearsal) reason-

ably well. Otherwise, the player will become distracted by the poor ensemble and not be able to demonstrate what he or she wants to hear. Thus there must be a neutral setting of parameters that allows the system to perform reasonably well "out of the box."

4.1. The timing model

We first consider a timing model for a single musical part. Our model is expressed in terms of two hidden sequences, $\{t_n\}$ and $\{s_n\}$ where t_n is the time, in seconds, of nth note onset and s_n is the tempo, in seconds per beat, for the nth note. These sequences evolve according to the model

$$s_{n+1} = s_n + \sigma_n \tag{3}$$

$$t_{n+1} = t_n + l_n s_n + \tau_n \tag{4}$$

where l_n is the length of the nth event, in beats.

With the "update" variables, $\{\sigma_n\}$ and $\{\tau_n\}$, set to 0, this model gives a literal and robotic musical performance with each inter-onset-interval, $t_{n+1} - t_n$, consuming an amount of time proportional to its length in beats, l_n. The introduction of the update variables allows time-varying tempo through the $\{\sigma_n\}$, and elongation or compression of note lengths with the $\{\tau_n\}$. We further assume that the $\{(\sigma_n, \tau_n)^t\}$ are independent with $(\sigma_n, \tau_n)^t \sim N(\mu_n, \Gamma_n)$, $n = 1, 2, \ldots$, and $(s_0, t_0)^t \sim N(\mu_0, \Gamma_0)$, thus leading to a joint Gaussian model on all model variables. The rhythmic interpretation embodied by the model is expressed in terms of the $\{\mu_n, \Gamma_n\}$ parameters. In this regard, the $\{\mu_n\}$ vectors represent the *tendencies* of the performance—where the player tends to speed up ($\sigma_n < 0$), slow down ($\sigma_n > 0$), and stretch ($\tau_n > 0$), while the $\{\Gamma_n\}$ matrices capture the repeatability of these tendencies.

It is simplest to think of Equations 3 and 4 as a timing model for single musical part. However, it is just as reasonable to view these equations as a timing model for the *composite* rhythm of the solo *and* orchestra. That is, consider the situation, depicted in Figure 3, in which the solo, orchestra, and composite rhythms have the following musical times (in beats):

solo	0	1/3		2/3	1	4/3		5/3	2
accomp	0		1/2		1		3/2		2
comp.	0	1/3	1/2	2/3	2	4/3	3/2	5/3	2

The $\{l_n\}$ for the composite would be found by simply taking the differences of rational numbers forming the composite rhythm: $l_1 = 1/3$, $l_2 = 1/6$, etc. In what follows, we regard Equations 3 and 4 as a model for this composite rhythm of the solo and orchestra parts.

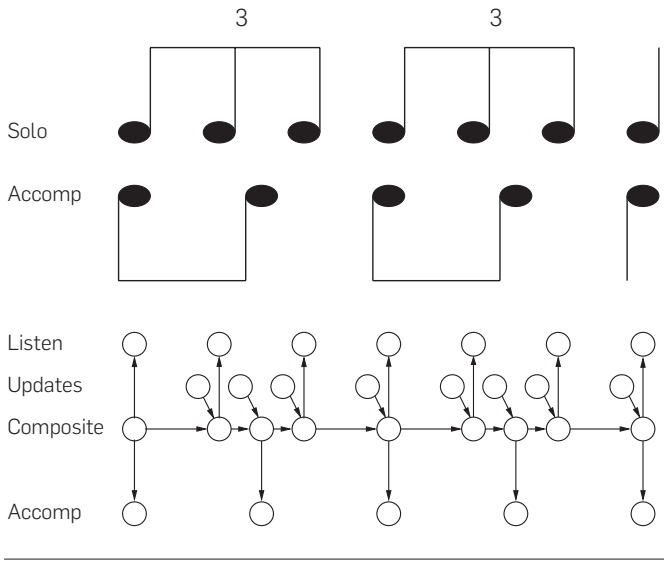
The *observable* variables in this model are the solo note onset estimates produced by Listen and the *known* note onsets of the orchestra (our system constructs these during the performance). Suppose that n indexes the events in the composite rhythm having associated solo notes, estimated by the $\{\hat{t}_n\}$. Additionally, suppose that n′ indexes the events having associated orchestra notes with onset times, $\{o_{n'}\}$. We model

$$\hat{t}_n = t_n + \epsilon_n$$

$$o_{n'} = t_{n'} + \delta_{n'}$$

where $\epsilon_n \sim N(0, \rho_s^2)$ and $\delta_{n'} \sim N(0, \rho_o^2)$. The result is the Gaussian graphical model depicted in the bottom panel of Figure 3. In this figure, the row labeled "Composite" corresponds to the $\{(s_n, t_n)\}$ variables of Equations 3 and 4, while the row labeled "Updates" corresponds to the $\{(\sigma_n, \tau_n)\}$ variables. The "Listen" row is the collection of estimated solo note onset times, $\{\hat{t}_n\}$ while the "Accompaniment" row corresponds to the orchestra times, $\{o_{n'}\}$.

Figure 3. Top: Two musical parts generate a *composite* rhythm when superimposed. Bot: The resulting graphical model arising from the composite rhythm.



4.2. The model in action

With the model in place we are now ready for real-time accompaniment. For our first rehearsal we initialize the model so that $\mu_n = 0$ for all n. This assumption in no way precludes our system from correctly interpreting and following tempo changes or other rhythmic nuances of the soloist. Rather, it states that, whatever we have seen so far in a performance, we *expect* future timing to evolve according to the current tempo.

In real-time accompaniment, our system is concerned only with scheduling the currently pending orchestra note time, $o_{n'}$. The time of this note is initially scheduled when we play the previous orchestra note, $o_{n'-1}$. At this point we compute the new mean of $o_{n'}$, conditioning on $o_{n'-1}$ and whatever other variables have been observed, and schedule $o_{n'}$ accordingly. While we wait for the currently scheduled time to occur, the Listen module may detect various solo events, $\hat{t}_n$. When this happens we *recompute* the mean of $o_{n'}$, conditioning on this new information. Sooner or later the actual clock time will catch up to the currently scheduled time of the n'th event, at which point the orchestra note is played. Thus an orchestra note may be rescheduled many times before it is actually played.

A particularly instructive example involves a run of many

solo notes culminating in a point of coincidence with the orchestra. As each solo note is detected we refine our estimate of the desired point of coincidence, thus gradually "honing in" on this point of arrival. It is worth noting that very little harm is done when Listen fails to detect a solo note. We simply predict the pending orchestra note conditioning on the variables we *have* observed.

The web page given before contains a video demonstrating this process. The video shows the estimated solo times from our score follower appearing as green marks on a spectrogram. Predictions of our accompaniment system are shown as analogous red marks. One can see the pending orchestra time "jiggling" as new solo notes are estimated, until finally the currently predicted time passes. In the video, one can see occasional solo notes that are never marked with green lines. These are notes for which the posterior onset time was not sufficiently peaked to merit a note detection. This happens most often with repeated pitches, for which our data model is less informative, and notes following longer notes, where our prior model is less opinionated. We simply treat such notes as unobserved and base our predictions only on the observed events.

The role of Predict is to "schedule" accompaniment notes, but what does this really mean in practice? Recall that our program plays audio by phase-vocoding (time-stretching) an orchestra-only recording. A time-frequency representation of such an audio file for the first movement of the Dvoŕák Cello concerto is shown in Figure 4. If you know the piece, you will likely be able to follow this spectrogram. In preparing this audio for our accompaniment system, we perform an off-line score alignment to determine where the various orchestra notes occur, as marked with vertical lines in the figure. Scheduling a note simply means that we change the phase-vocoder's play rate so that it arrives at the appropriate audio file position (vertical line) at the scheduled time. Thus the play rate is continually modified as the performance evolves. This is our only "control" the orchestra performance.

After one or more "rehearsals," we *adapt* our timing model to the soloist to better anticipate future performances. To do this, we first perform an off-line estimate of the solo note times using Equation 2, only conditioning on the *entire* sequence of frames, $y_1, \ldots, y_T$, using the forward–backward algorithm to identify the most likely onset time for each note. Using one or more such rehearsals, we can iteratively reestimate the model parameters $\{\mu_n\}$ using the EM algorithm, resulting in both measurable and perceivable improvement of prediction accuracy. While, in principle, we can also estimate the $\{\Gamma_n\}$ parameters, we have observed little or no benefit from doing so.

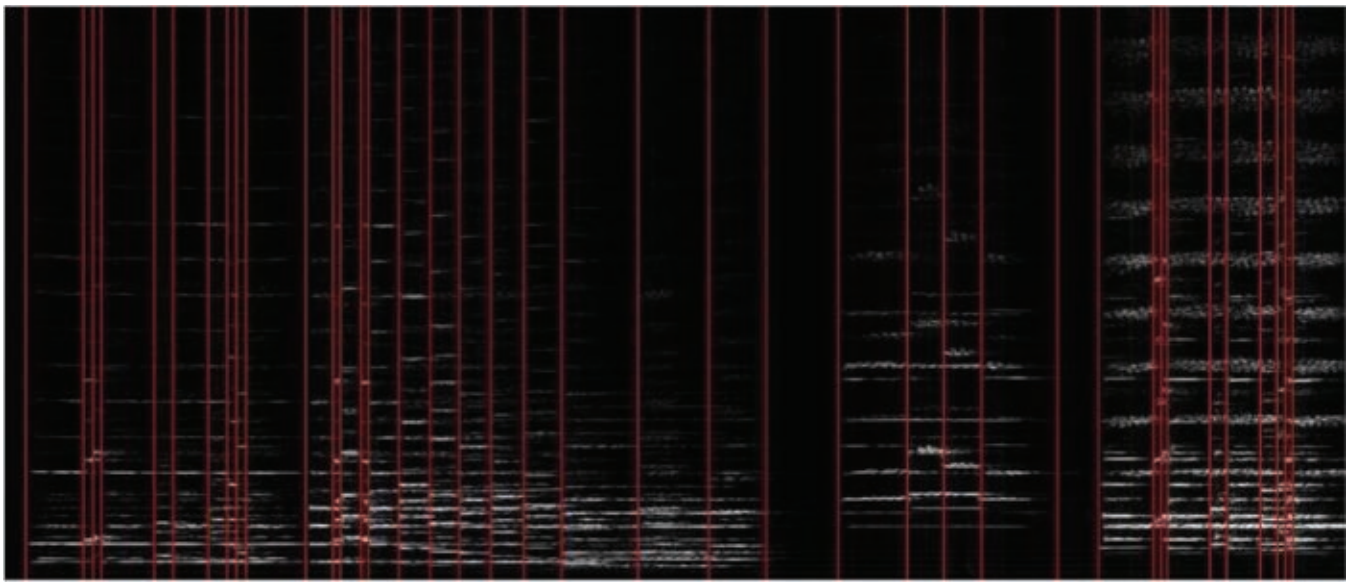
In practice, we have found the soloist's interpretation to be something of a "moving target." At first this is because the soloist tends to compromise somewhat in the initial rehearsals, pulling the orchestra in the desired *direction*, while not actually reaching the target interpretation. But even after the soloist seems to settle down to a particular interpretation on a given day, we often observe further "interpretation drift" over subsequent meetings. Of course, without this drift one's ideas could never improve! For this reason we train the model using the most recent several rehearsals, thus facilitating the continually evolving nature of musical interpretation.

5. MUSICAL EXPRESSION AND MACHINE LEARNING

Our system learns its musicality through "osmosis." If the soloist plays in a musical way, and the orchestra manages to closely follow the soloist, then we hope the orchestra will *inherit* this musicality. This manner of learning by imitation works well in the concerto setting, since the division of authority between the players is rather extreme, mostly granting the "right of way" to the soloist.

In contrast, the pure following approach is less reasonable when the accompaniment needs a sense of musicality that acts independently, or perhaps even in opposition, to what

Figure 4. A "spectrogram" of the opening of the first movement of the Dvoŕák Cello concerto. The horizontal axis of the figure represents time while the vertical axis represents frequency. The vertical lines show the note times for the orchestra.



other players do. Such a situation occurs with the early-stage accompaniment problem discussed in Section 1, as here one cannot learn the desired musicality from the live player. Perhaps the accompaniment *antithesis* of the concerto setting is the opera orchestra, in which the "accompanying" ensemble is often on equal footing with the soloists. We observed the nadir of our system's performance in an opera rehearsal where our system served as rehearsal pianist. What these two situations have in common is that they require an accompanist with independent musical knowledge and goals.

How can we more intelligently model this musicality? An incremental approach would begin by observing that our timing model of Equations 3 and 4 is over-parametrized, with more degrees of freedom than there are notes. We make this modeling choice because we do not know which degrees of freedom are needed ahead of time, so we use the training data from the soloist to help sort this out. Unnecessary learned parameters may contribute some noise to the resulting timing model, but the overall result is acceptable.

One possible line of improvement is simply decreasing the model's freedom—surely the player does not wish to change the tempo *and* apply tempo-independent note length variation on every note. For instance, one alternative model adds a hidden discrete process that "chooses," for each note, between three possibilities: variation of *either* tempo *or* note length, or no variation of either kind. Of these, the choice of neither variation would be the most likely a priori, thus biasing the model toward simpler musical interpretations. The resulting model is a Switching Kalman Filter.[15] While exact inference is no longer possible with such a model, we expect that one can make approximations that will be good enough to realize the full potential of the model.

Perhaps a more ambitious approach analyzes the musical score itself to choose the locations requiring degrees of freedom. One can think of this approach as adding "joints" to the musical structure so that it deforms into musically reasonable shapes as a musician applies external force. Here there is an interesting connection with the work on expressive synthesis, such as Widmer and Goebl,[16] in which one algorithmically constructs an expressive rendition of a previously unseen piece of music, using ideas of machine learning. One approach here associates various score situations, defined in terms of local configurations of score features, with interpretive actions. The associated interpretive actions are learned by estimating timing and loudness parameters from a performance corpus, over all "equivalent" score locations. Such approaches are far more ambitious than our present approach to musicality, as they try to understand expression *in general*, rather than in a specific musical context.

The understanding and synthesis of musical expression is one of the most interesting music-science problems, and while progress has been achieved in recent years, it would still be fair to call the problem "open." One of the principal challenges here is that one cannot directly map observable surface-level attributes of the music, such as pitch contour or local rhythm context, into interpretive actions, such as delay, or tempo or loudness change. Rather, there is a murky intermediate level in which the musician comes to some understanding of the musical *meaning*, on which the interpretive decisions are based. This meaning comes from several different aspects of the music. For example, some comes from musical structure, as in the way one might slow down at the end of a phrase, giving a sense of musical closure. Some meaning comes from prosodic aspects, analogous to speech, such as a local point of arrival, which may be maybe emphasized or delayed. A third aspect of meaning describes an overall character or *affect* of a section of music, such as excited or calm. While there is no official taxonomy of musical interpretation, most discussions on this subject revolve around intermediate identifications of this kind, and the interpretive actions they require.[10]

From the machine learning point of view, it is impossible to learn anything useful from a single example, thus one must group together many examples of the same musical situation in order to learn their associated interpretive actions. Thus it seems natural to model the music in terms of some latent variables that implicitly categorize individual notes or sections of music. What should the latent variables be, and how can one describe the dependency structure among them? While we cannot answer these questions, we see in them a good deal of depth and challenge, and recommend this problem to the musically inclined members of the readership with great enthusiasm.

Acknowledgments

This work was supported by NSF Grants IIS-0812244 and IIS-0739563.

References

1. Cemgil, A.T., Kappen, H.J., Barber, D. A generative model for music transcription. *IEEE Trans. Audio Speech Lang. Process. 14*, 2 (Mar. 2006), 679–694.
2. Cont, A., Schwarz, D., Schnell, N. From Boulez to ballads: Training ircam's score follower. In *Proceedings of the International Computer Music Conference* (2005), 241–248.
3. Dannenberg, R., Mont-Reynaud, B. Following an improvisation in real time. In *Proceedings of the 1987 International Computer Music Conference* (1987), 241–248.
4. Dannenberg, R., Mukaino, H. New techniques for enhanced quality of computer accompaniment. In *Proceedings of the 1988 International Computer Music Conference* (1988), 243–249.
5. Flanagan, J.L., Golden, R.M. Phase vocoder. *Bell Syst. Tech. J. 45* (Nov. 1966), 1493–1509.
6. Franklin, J. Improvisation and learning. In *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.
7. Klapuri, A., Davy, M. (editors). *Signal Processing Methods for Music Transcription*. Springer-Verlag, New York, 2006.
8. Lippe, C. Real-time interaction among composers, performers, and computer systems. *Inf. Process. Soc. Jpn.* SIG Notes, *123* (2002), 1–6.
9. Pachet, F. Beyond the cybernetic jam fantasy: The continuator. *IEEE Comput. Graph. Appl. 24*, 1 (2004), 31–35.
10. Palmer, C. Music performance. *Annu. Rev. Psychol. 48* (1997), 115–138.
11. Raphael, C. A Bayesian network for real-time musical accompaniment. In *Advances in Neural Information Processing Systems* (NIPS) 14. MIT Press, 2002.
12. Rowe, R. *Interactive Music Systems*. MIT Press, 1993.
13. Sagayama, T.N.S., Kameoka, H. Specmurt anasylis: A piano-roll-visualization of polyphonic music signal by deconvolution of log-frequency spectrum. In *Proceedings 2004 ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA2004)* (2004).
14. Schwarz, D. Score following commented bibliography, 2003.
15. Shumway, R.H., Stoffer, D.S. Dynamic linear models with switching. *J. Am. Stat. Assoc. 86* (1991), 763–769.
16. Widmer, G., Goebl, W. Computational models for expressive music performance: The state of the art. *J. New Music Res. 33*, 3 (2004), 203–216

Christopher Raphael (craphael@indiana.edu), School of Informatics and Computing, Indiana University, Bloomington, IN.

DOI:10.1145/1897852.1897876

Technical Perspective
VL2

By Jennifer Rexford

THE INTERNET IS increasingly a platform for online services—such as email, Web search, social networks, and virtual worlds—running on rack after rack of servers in data centers. The servers not only communicate with end users, but also with each other to analyze data (for example, to build a search index) or compose Web pages (for example, by combining data from multiple backend servers). With the advent of large data centers, the study of the networks that interconnect these servers has become an important topic to researchers and practitioners alike.

Data-center networking presents unique opportunities and challenges, compared to traditional backbone and enterprise networks:

- In a data center, the same company controls both the servers and the network elements, enabling new network architectures that implement key functionality on the end-host computers.
- Servers are installed in fixed units, such as racks or even trucks filled with racks driven directly into the data center. This leads to very uniform wiring topologies, such as fat trees or Clos networks, reminiscent of the massively parallel computers designed in the 1990s.
- The traffic load in data centers is often quite heavy and non-uniform, due to new backend applications like MapReduce; the traffic can also be quite volatile, varying dramatically and unpredictably over time.

In light of these new characteristics, researchers have been revisiting everything in networking—from addressing and congestion control to routing and the underlying topology—with the unique needs of data centers in mind.

The following paper presents one of the first measurement studies of network traffic in data centers, highlighting specifically the volatility of the traffic even on a relatively small timescale. These observations led the authors to design an "agile" network engineered for all-to-all connectivity with no contention inside the network. This gives data-center operators the freedom to place applications on any servers, without concern for the performance of the underlying network. Having an agile network greatly simplifies the task of designing and running online services.

More generally, the authors propose a simple abstraction—a single "virtual" layer-two switch (hence the name "VL2") for each service, with no interference with the many other services running in the same data center. They achieve this goal through several key design decisions, including flat addressing (so service instances can run on any server, independent of its location) and Valiant Load Balancing (to spread traffic uniformly over the network). A Clos topology ensures the network has many paths between each pair of servers. To scale to large data centers, the servers take responsibility for translating addresses to the appropriate "exit point" from the network, obviating the need for the networking equipment to keep track of the many end hosts in the data center.

This paper is a great example of rethinking networking from scratch, while coming full circle to work with today's equipment.

In addition to proposing an effective design, the authors illustrate how to build the solution using mechanisms available in existing network switches (for example, equal-cost multipath routing, IP anycast, and packet encapsulation). This allows data centers to deploy VL2 with no changes to the underlying switches, substantially lowering the barrier for practical deployment. This paper is a great example of rethinking networking from scratch, while coming full circle to work with today's equipment. Indeed, the work depicted in the VL2 paper has already spawned substantial follow-up work in the networking research community, and likely will for years to come.

Jennifer Rexford is a professor in the Department of Computer Science at Princeton University, Princeton, NJ.

DOI:10.1145/1897852.1897877

VL2: A Scalable and Flexible Data Center Network

By Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta

Abstract

To be agile and cost effective, data centers must allow dynamic resource allocation across large server pools. In particular, the data center network should provide a simple flat abstraction: it should be able to take any set of servers anywhere in the data center and give them the illusion that they are plugged into a physically separate, noninterfering Ethernet switch with as many ports as the service needs. To meet this goal, we present VL2, a practical network architecture that scales to support huge data centers with uniform high capacity between servers, performance isolation between services, and Ethernet layer-2 semantics. VL2 uses (1) flat addressing to allow service instances to be placed anywhere in the network, (2) Valiant Load Balancing to spread traffic uniformly across network paths, and (3) end system–based address resolution to scale to large server pools without introducing complexity to the network control plane. VL2's design is driven by detailed measurements of traffic and fault data from a large operational cloud service provider. VL2's implementation leverages proven network technologies, already available at low cost in high-speed hardware implementations, to build a scalable and reliable network architecture. As a result, VL2 networks can be deployed today, and we have built a working prototype. We evaluate the merits of the VL2 design using measurement, analysis, and experiments. Our VL2 prototype shuffles 2.7 TB of data among 75 servers in 395 s—sustaining a rate that is 94% of the maximum possible.

1. INTRODUCTION

Cloud services are driving the creation of huge data centers, holding tens to hundreds of thousands of servers, that concurrently support a large and dynamic number of distinct services (web apps, e-mail, map-reduce clusters, etc.). The case for cloud service data centers depends on a scale-out design: reliability and performance achieved through large pools of resources that can be rapidly reassigned between services as needed. With data centers being built with over 100,000 servers, at an amortized cost approaching $12 million per month,[14] the most desirable property for a data center is *agility*—the ability to assign any server to any service. Anything less inevitably results in stranded resources and wasted money.

Unfortunately, the data center network is not up to the task, falling short in several ways. First, existing architectures do not provide enough capacity between the servers they interconnect. Conventional architectures rely on treelike network configurations built from expensive hardware. Due to the high equipment cost, the capacity between different levels of the tree is typically oversubscribed by factors of 1:5 or more, with paths near the root oversubscribed by factors of 1:80 to 1:240. This oversubscription limits communication between servers to the point that it fragments the server pool—congestion and computation hot spots are prevalent even when spare capacity is available elsewhere. Second, while data centers host multiple services, the network does little to prevent a traffic flood in one service from affecting other services around it—when one service experiences a traffic flood, it is common for all those sharing the same network subtree to suffer collateral damage. Third, the routing design in conventional networks achieves scale by assigning servers topologically significant IP addresses and dividing servers up among VLANs. However, this creates an enormous configuration burden when servers must be reassigned among services, further fragmenting the resources of the data center. The human involvement typically required in these reconfigurations undermines speed of deployment.

To overcome these limitations in the current network and achieve agility, we arrange for the network to implement a familiar and concrete model: give each service the illusion that all the servers assigned to it, and only those servers, are connected by a single noninterfering Ethernet switch—a *Virtual Layer 2*—and maintain this illusion even as the size of each service varies from 1 server to 100,000. Realizing this vision for the data center network concretely translates into building a network that meets the following three objectives:

- **Uniform high capacity**: The maximum rate of a server-to-server traffic flow should be limited only by the available capacity on the network-interface cards of the sending and receiving servers, and it should be possible to assign servers to a service without having to consider network topology.
- **Performance isolation**: Traffic of one service should not be affected by the traffic of any other service, just as if each service was connected by a separate physical switch.
- **Layer-2 semantics**: The servers in each service should experience the network as if it were an Ethernet Local Area Network (LAN). Data center management soft-

A previous version of this paper was published in the *Proceedings of SIGCOMM '09* (Barcelona, Spain, Aug. 17–21, 2009). ACM, New York.

ware should be able to assign any IP address the service requests to any server, and virtual machines should be able to migrate to any server while keeping the same IP address. Finally, features like link-local broadcast, on which many legacy applications depend, should work.

We design, implement, and evaluate VL2, a network architecture for data centers that meets these three objectives and thereby achieves agility.

Design philosophy: In designing VL2, a primary goal was to create a network architecture that could be deployed today, so we limit ourselves from making any changes to the hardware of the switches or servers, and we require that legacy applications work unmodified. Our approach is to build a network that operates like a very large switch—choosing simplicity and high performance over other features when needed. We sought to use robust and time-tested control plane protocols, and we avoid adaptive routing schemes that might theoretically offer more bandwidth but open thorny problems that might not need to be solved and would take us away from vanilla, commodity, high-capacity switches.

We observe, however, that the software and operating systems on data center servers are already extensively modified (e.g., to create hypervisors for virtualization or blob file systems to store data across servers). Therefore, VL2's design explores a new split in the responsibilities between host and network—using a layer 2.5 shim in servers' network stack to work around limitations of the network devices. No new switch software or switch APIs are needed.

Topology: VL2 consists of a network built from low-cost switch ASICs arranged into a Clos topology that provides extensive path diversity between servers. This design replaces today's mainframe-like large, expensive switches with broad layers of low-cost switches that can be scaled out to add more capacity and resilence to failure. In essence, VL2 applies the principles of RAID (redundant arrays of inexpensive disks) to the network.

Traffic engineering: Our measurements show data centers have tremendous volatility in their workload, their traffic, and their failure patterns. To cope with this volatility in the simplest manner, we adopt Valiant Load Balancing (VLB) to spread traffic across all available paths without any centralized coordination or traffic engineering. Using VLB, each server independently picks a path at random through the network for each of the flows it sends to other servers in the data center. Our experiments verify that using this design achieves both uniform high capacity and performance isolation.

Control plane: The switches that make up the network operate as layer-3 routers with routing tables calculated by OSPF, thereby enabling the use of multiple paths while using a time-tested protocol. However, the IP addresses used by services running in the data center must not be tied to particular switches in the network, or the ability for agile reassignment of servers between services would be lost. Leveraging a trick used in many systems,[9] VL2 assigns servers IP addresses that act as names alone, with no topological significance. When a server sends a packet, the shim layer on the server invokes a directory system to learn the actual location of the destination and then tunnels the original packet there. The shim layer also helps eliminate the scalability problems created by ARP in layer-2 networks, and the tunneling improves our ability to implement VLB. These aspects of the design enable VL2 to provide layer-2 semantics—eliminating the fragmentation and waste of server pool capacity that the binding between addresses and locations causes in the existing architecture.

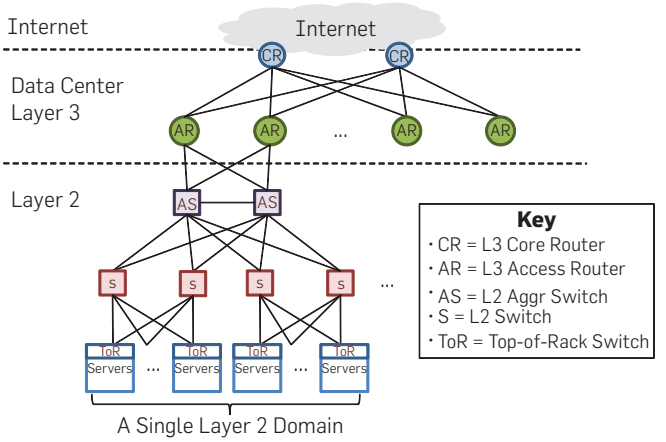
Contributions: In the course of this paper, we describe the current state of data center networks and the traffic across them, explaining why these are important to designing a new architecture. We present VL2's design, which we have built and deployed into an 80-server cluster. Using the cluster, we experimentally validate that VL2 has the properties set out as objectives, such as uniform capacity and performance isolation. We also demonstrate the speed of the network, such as its ability to shuffle 2.7TB of data among 75 servers in 395s (averaging 58.8Gbps). Finally, we describe our experience applying VLB in a new context, the inter-switch fabric of a data center, and show that VLB smooths utilization while eliminating persistent congestion.

2. BACKGROUND

In this section, we first explain the dominant design pattern for data center architecture today.[5] We then discuss why this architecture is insufficient to serve large cloud-service data centers.

As shown in Figure 1, the network is a hierarchy reaching from a layer of servers in racks at the bottom to a layer of core routers at the top. There are typically 20–40 servers per rack, each singly connected to a Top of Rack (ToR) switch with a 1Gbps link. ToRs connect to two aggregation switches for redundancy, and these switches aggregate further connecting to access routers. At the top of the hierarchy, core routers carry traffic between access routers and manage traffic into and out of the data center. All links use Ethernet as a physical-layer protocol, with a mix of copper and fiber cabling. All switches below each pair of access routers form a single layer-2 domain. The number of servers in a single

Figure 1. A conventional network architecture for data centers (adapted from figure by Cisco[5]).



layer-2 domain is typically limited to a few hundred due to Ethernet scaling overheads (packet flooding and ARP broadcasts). To limit these overheads and to isolate different services or logical server groups (e.g., e-mail, search, web front ends, web back ends), servers are partitioned into virtual LANs (VLANs) placed into distinct layer-2 domains. Unfortunately, this conventional design suffers from three fundamental limitations:

Limited server-to-server capacity: As we go up the hierarchy, we are confronted with steep technical and financial barriers in sustaining high bandwidth. Thus, as traffic moves up through the layers of switches and routers, the oversubscription ratio increases rapidly. For example, servers typically have 1:1 oversubscription to other servers in the same rack—that is, they can communicate at the full rate of their interfaces (e.g., 1 Gbps). We found that uplinks from ToRs are typically 1:2 to 1:20 oversubscribed (i.e., 1–10 Gbps of uplink for 20 servers), and paths through the highest layer of the tree can be 1:240 oversubscribed. This large oversubscription factor fragments the server pool by preventing idle servers from being assigned to overloaded services, and it severely limits the entire data center's performance.

Fragmentation of resources: As the cost and performance of communication depends on distance in the hierarchy, the conventional design encourages service planners to cluster servers nearby in the hierarchy. Moreover, spreading a service outside a single layer-2 domain frequently requires the onerous task of reconfiguring IP addresses and VLAN trunks, since the IP addresses used by servers are topologically determined by the access routers above them. Collectively, this contributes to the squandering of computing resources across the data center. The consequences are egregious. Even if there is plentiful spare capacity throughout the data center, it is often effectively reserved by a single service (and not shared), so that this service can scale out to nearby servers to respond rapidly to demand spikes or to failures. In fact, the growing resource needs of one service have forced data center operations to evict other services in the same layer-2 domain, incurring significant cost and disruption.

Poor reliability and utilization: Above the ToR, the basic resilience model is 1:1. For example, if an aggregation switch or access router fails, there must be sufficient remaining idle capacity on the counterpart device to carry the load. This forces each device and link to be run up to at most 50% of its maximum utilization. Inside a layer-2 domain, use of the Spanning Tree Protocol means that even when multiple paths between switches exist, only a single one is used. In the layer-3 portion, Equal Cost Multipath (ECMP) is typically used: when multiple paths of the same length are available to a destination, each router uses a hash function to spread flows evenly across the available next hops. However, the conventional topology offers at most two paths.

3. MEASUREMENTS AND IMPLICATIONS

Developing a new network architecture requires a quantitative understanding of the traffic matrix (who sends how much data to whom and when?) and churn (how often does the state of the network change due to switch/link failures and recoveries, etc.?). We studied the production data centers of a large cloud service provider, and we use the results to drive our choices in designing VL2. Details of the methodology and results can be found in other papers.[10, 16] Here we present the key findings that directly impact the design of VL2.

Most traffic is internal to the data center: The ratio of traffic volume between servers in our data centers to traffic entering/leaving our data centers is currently around 4:1 (excluding CDN applications). An increasing fraction of the computation in data centers involves back-end computations, and these are driving the demands for network bandwidth.

The network bottlenecks computation: Data center computation is focused where high-speed access to data on memory or disk is fast and cheap. Even inside a single data center, the network is a bottleneck to computation—we frequently see switches whose uplinks are above 80% utilization. Intense computation and communication on data does not straddle data centers due to the cost of long-haul links.

Structured flow sizes: Figure 2 illustrates the nature of flows within the monitored data center. The flow size statistics (marked as "+"s) show that the majority of flows are small (a few KB); most of these small flows are hellos and meta-data requests to the distributed file system. To examine longer flows, we compute a statistic termed *total bytes* (marked as "o"s) by weighting each flow size by its number of bytes. Total bytes tells us, for a random byte, the distribution of the flow size it belongs to. Almost all the bytes in the data center are transported in flows whose lengths vary from about 100MB to about 1GB. The mode at around 100MB springs from the fact that the distributed file system breaks long files into 100-MB-long chunks. Importantly, there are almost no flows over a few GB.

Figure 3 shows the probability density function (as a fraction of time) for the number of concurrent flows going in and out of a machine. There are two modes. More

Figure 2. Mice are numerous; 99% of flows are smaller than 100MB. However, more than 90% of bytes are in flows between 100MB and 1GB.

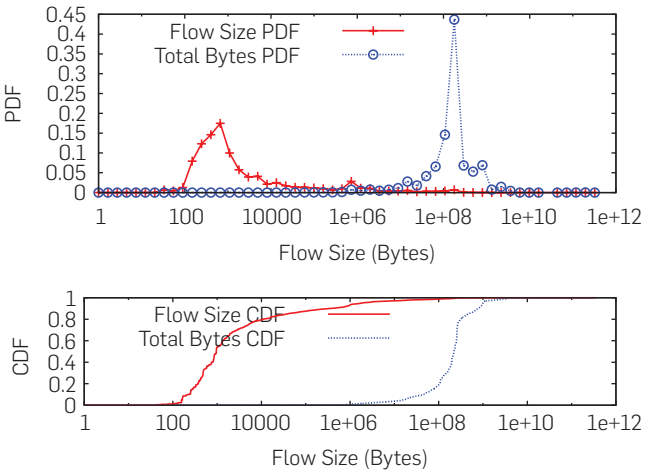
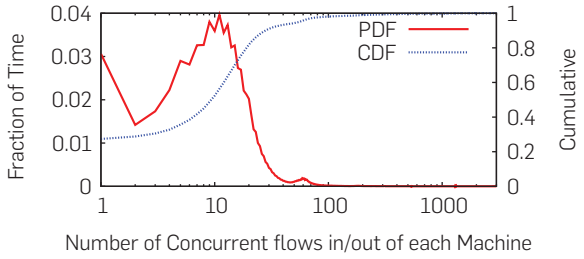


Figure 3. Number of concurrent connections has two modes: (1) 10 flows per node more than 50% of the time and (2) 80 flows per node for at least 5% of the time.



than 50% of the time, an average machine has about ten concurrent flows, but at least 5% of the time it has greater than 80 concurrent flows. We almost never see more than 100 concurrent flows.

The distributions of flow size and number of concurrent flows both imply that flow-based VLB will perform well on this traffic. Since even big flows are only 100MB (1s of transmit time at 1Gbps), randomizing at flow granularity (rather than packet) will not cause perpetual congestion if there is unlucky placement of too many flows in the same link.

Volatile traffic patterns: While the sizes of flows show a strong pattern, the traffic patterns inside a data center are highly divergent. When we cluster the traffic patterns, we find that more than 50 representative patterns are required to describe the traffic in the data center. Further, the traffic pattern varies frequently—60% of the time the network spends only 100 s in one pattern before switching to another.

Frequent failures: As discussed in Section 2, conventional data center networks apply 1 + 1 redundancy to improve reliability at higher layers of the hierarchical tree. This hierarchical topology is intrinsically unreliable—even with huge effort and expense to increase the reliability of the network devices close to the top of the hierarchy, we still see failures on those devices resulting in significant downtime. In 0.3% of failures, all redundant components in a network device group became unavailable (e.g., the pair of switches that comprise each node in the conventional network (Figure 1) or both the uplinks from a switch). The main causes of failures are network misconfigurations, firmware bugs, and faulty components.

With no obvious way to eliminate failures from the top of the hierarchy, VL2's approach is to broaden the top levels of the network so that the impact of failures is muted and performance degrades gracefully, moving from 1 + 1 redundancy to $n + m$ redundancy.

4. VIRTUAL LAYER 2 NETWORKING

Before detailing our solution, we briefly discuss our design principles and preview how they will be used in our design.

Randomizing to cope with volatility: The high divergence and unpredictability of data center traffic matrices suggest that optimization-based approaches to traffic engineering risk congestion and complexity to little benefit. Instead, VL2 uses VLB: destination-independent (e.g., random) traffic spreading across the paths in the network. VLB, in theory, ensures a *noninterfering* packet-switched network[6] (the counterpart of a non-blocking circuit-switched network) as long as (a) traffic spreading ratios are uniform, and (b) the offered traffic patterns do not violate edge constraints (i.e., line card speeds). We use ECMP to pursue the former and TCP's end-to-end congestion control to pursue the latter. While these design choices do not perfectly ensure the two assumptions (a and b), we show in Section 5.1 that our scheme's performance is close to the optimum in practice.

Building on proven networking technology: VL2 is based on IP routing and forwarding technologies already available in commodity switches: link-state routing, ECMP forwarding, and IP any-casting. VL2 uses a link-state routing protocol to maintain the switch-level topology, but not to disseminate end hosts' information. This strategy protects switches from needing to learn voluminous, frequently changing host information. Furthermore, the routing design uses ECMP forwarding along with anycast addresses to enable VLB while minimizing control plane messages and churn.

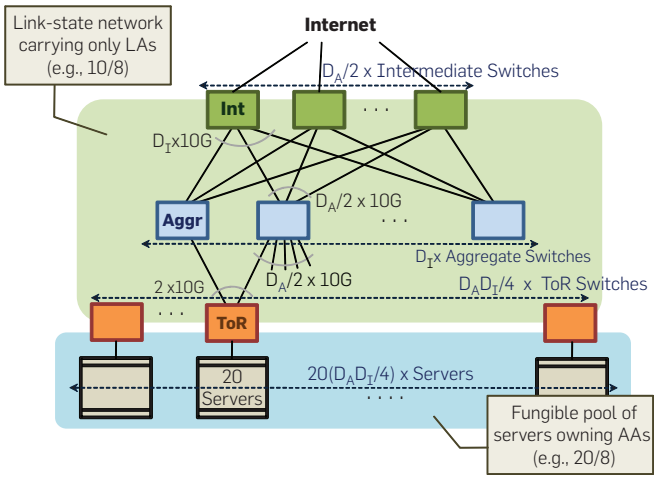
Separating names from locators: To be able to rapidly grow or shrink server allocations and rapidly migrate VMs, the data center network must support agility, which means support hosting any service on any server. This, in turn, calls for separating names from locations. VL2's addressing scheme separates servers' names, termed application-specific addresses (AAs), from their locations, termed location-specific addresses (LAs). VL2 uses a scalable, reliable directory system to maintain the mappings between names and locators. A shim layer running in the networking stack on every host, called the VL2 agent, invokes the directory system's resolution service.

Embracing end systems: The rich and homogeneous programmability available at data center hosts provides a mechanism to rapidly realize new functionality. For example, the VL2 agent enables fine-grained path control by adjusting the randomization used in VLB. The agent also replaces Ethernet's ARP functionality with queries to the VL2 directory system. The directory system itself is also realized on regular servers, rather than switches, and thus offers flexibility, such as fine-grained access control between application servers.

4.1. Scale-out topologies

As described in Sections 2 and 3, conventional hierarchical data center topologies have poor bisection bandwidth and are susceptible to major disruptions due to device failures. Rather than *scale up* individual network devices with more capacity and features, we *scale out* the devices—building a broad network offering huge *aggregate* capacity using a large number of simple, inexpensive devices, as shown in Figure 4. This is an example of a folded Clos network[6] where the links between the intermediate switches and the aggregation switches form a complete bipartite graph. As in the conventional topology, ToRs connect to two aggregation switches, but the large number of paths between any two aggregation switches means that if there are n intermediate switches, the failure of any one of them reduces the bisection bandwidth by only $1/n$—a desirable property we call *graceful degradation*

Figure 4. An example Clos network between aggregation and intermediate switches provides a richly connected backbone well suited for VLB. The network is built with two separate address families—topologically significant locator-specific addresses (LAs) and flat application-specific addresses (AAs).



of bandwidth. Further, it is easy and inexpensive to build a Clos network for which there is no oversubscription (further discussion on cost is given in Section 6). For example, in Figure 4, we use D_A-port Aggregation and D_I-port intermediate switches, and connect these switches such that the capacity between each layer is $D_I D_A/2$ times the link capacity.

The Clos topology is exceptionally well suited for VLB in that by forwarding traffic through an intermediate switch that is chosen in a destination-independent passion (e.g., randomly chosen), the network can provide bandwidth guarantees for any traffic matrices that obey the hose model.[8] Meanwhile, routing remains simple and resilient on this topology—take a random path up to a random intermediate switch and a random path down to a destination ToR switch.

4.2. VL2 addressing and routing

This section explains the motion of packets in a VL2 network, and how the topology, routing design, VL2 agent, and directory system combine to virtualize the underlying network fabric and create the illusion that hosts are connected to a big, noninterfering data center–wide layer-2 switch.

Address Resolution and Packet Forwarding: VL2 uses two separate classes of *IP-address* illustrated in Figure 4. The network infrastructure operates using LAs; all switches and interfaces are assigned LAs, and switches run an IP-based (layer-3) link-state routing protocol that disseminates only these LAs. This allows switches to obtain complete knowledge about the switch-level topology, as well as forward any packets encapsulated with LAs along the shortest paths. On the other hand, applications use permanent AAs, which remain unaltered no matter how servers' locations change due to VM migration or reprovisioning. Each AA (server) is associated with an LA, the IP address of the ToR switch to which the application server is connected. The ToR switch need not be physical hardware—it could be a virtual switch or hypervisor implemented in software on the server itself!

The VL2 directory system stores the mapping of AAs to LAs, and this mapping is created when application servers are provisioned to a service and assigned AA addresses. Resolving these mappings through a unicast-based custom protocol eliminates the ARP and DHCP scaling bottlenecks that plague large Ethernets.

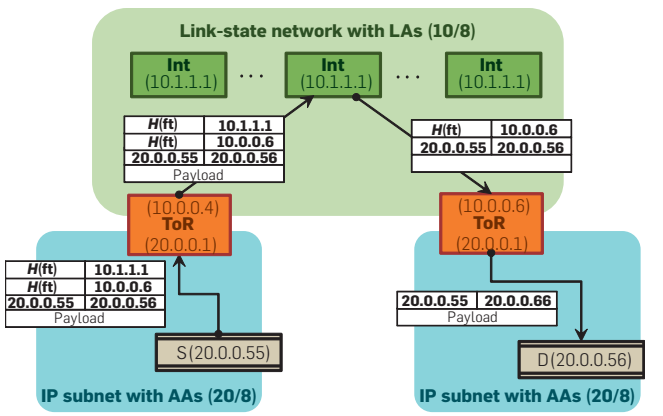
Packet forwarding: Since AA addresses are not announced into the routing protocols of the network, for a server to receive a packet the sending server must first encapsulate the packet (Figure 5), setting the destination of the outer header to the LA of the destination AA. Once the packet arrives at the LA (the destination ToR or hypervisor), the switch (the ToR or the VM switch in the destination hypervisor) decapsulates the packet and delivers it to the destination AA given in the inner header.

Address resolution and access control: Servers in each service are configured to believe that they all belong to the same IP subnet, so when an application sends a packet to an AA for the first time, the networking stack on the host generates a broadcast ARP request for the destination AA. The VL2 agent running in the source's networking stack intercepts the ARP request and converts it to a unicast query to the VL2 directory system. The directory system answers the query with the LA of the ToR to which packets should be tunneled. During this resolution process, the directory server can additionally evaluate the access-control policy between the source and destination and selectively reply to the resolution query, enforcing necessary isolation policies between applications.

These addressing and forwarding mechanisms were chosen for two main reasons. First, they make it possible to use low-cost switches, which often have small routing tables (typically just 16K entries) that can hold only LA routes, without concern for the huge number of AAs. Second, they allow the control plane to support agility with very little overhead; the design obviates frequent link-state advertisements to disseminate host-state changes and host/switch reconfiguration.

Random Traffic Spreading over Multiple Paths: To offer hot

Figure 5: VLB in an example VL2 network. Sender `S` sends packets to destination `D` via a randomly chosen intermediate switch using IP-in-IP encapsulation. AAs are from `20/8`, and LAs are from `10/8`. *H*(`ft`) denotes a hash of the five tuple.



spot–free performance for arbitrary traffic matrices, VL2 uses VLB as its traffic engineering philosophy. As illustrated in Figure 5, VL2 achieves VLB using a combination of ECMP routing implemented by the switches and packet encapsulation implemented by the shim on each server. ECMP, a mechanism already implemented in the hardware of most switches, will distribute flows across the available paths in the network, with the packets with the same source and destination address taking the same path to avoid packet reordering. To leverage all the available paths in the network and overcome some limitations in ECMP, the VL2 agent on each sender encapsulates each packet to an intermediate switch. Hence, the packet is first delivered to one of the intermediate switches, decapsulated by the switch, delivered to the ToR's LA, decapsulated again, and finally sent to the destination server. The source address in the outer headers of the encapsulated packet is set to a hash of the inner packet's addresses and ports—this provides additional entropy to better distribute flows between the same servers across the available paths.

One potential issue for both ECMP and VLB is the chance that uneven flow sizes and random spreading decisions will cause transient congestion on some links. Our evaluation did not find this to be a problem on data center workloads (Section 5), but should it occur, the VL2 agent on the sender can detect and deal with it via simple mechanisms. For example, it can change the hash used to create the source address periodically or whenever TCP detects a severe congestion event (e.g., a full window loss) or an Explicit Congestion Notification.

4.3. Maintaining host information using the VL2 directory system

The VL2 directory system provides two key functions: (1) *lookups* and *updates* for AA-to-LA mappings and (2) a reactive cache update mechanism that ensures eventual consistency of the mappings with very little update overhead (Figure 6).

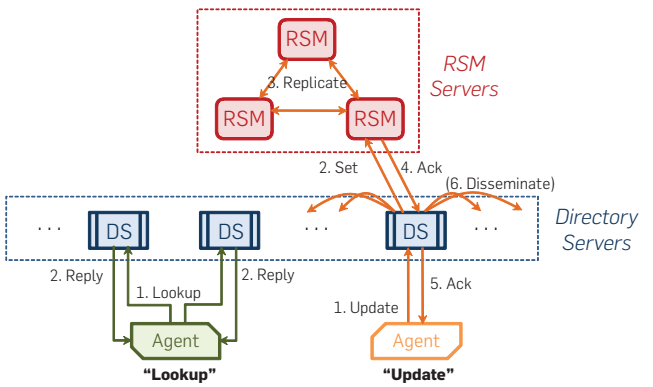
We expect the lookup workload for the directory system to be frequent and bursty because servers can communicate with up to hundreds of other servers in a short time period, with each new flow generating a lookup for an AA-to-LA mapping. The bursty nature of workload implies that lookups require high throughput and low response time to quickly establish a large number of connections. Since lookups replace ARP, their response time should match that of ARP, that is, tens of milliseconds. For updates, however, the workload is driven by server-deployment events, most of which are planned ahead by the data center management system and hence can be batched. The key requirement for updates is reliability, and response time is less critical.

Our directory service replaces ARP in a conventional L2 network, and ARP ensures eventual consistency via timeout and broadcasting. This implies that eventual consistency of AA-to-LA mappings is acceptable as long as we provide a reliable update mechanism. Nonetheless, we intend to support live VM migration in a VL2 network; our directory system should be able to correct all the stale entries without breaking any ongoing communications.

The differing performance requirements and workload patterns of lookups and updates lead us to a two-tiered directory system architecture consisting of (1) a modest number (50–100 servers for 100K servers) of read-optimized, replicated lookup servers that cache AA-to-LA mappings and that communicate with VL2 agents, and (2) a small number (5–10 servers) of write-optimized, asynchronous replicated state-machine (RSM) servers offering a strongly consistent, reliable store of AA-to-LA mappings. The lookup servers ensure low latency, high throughput, and high availability for a high lookup rate. Meanwhile, the RSM servers ensure strong consistency and durability for a modest rate of updates using the Paxos[19] consensus algorithm.

Each lookup server caches all the AA-to-LA mappings stored at the RSM servers and independently replies to lookup queries from agents using the cached state. Since strong consistency is not required, a lookup server lazily synchronizes its local mappings with the RSM every 30s. To achieve high availability and low latency, an agent sends a query to k (two in our prototype) randomly chosen lookup servers and simply chooses the fastest reply. Since AA-to-LA mappings are cached at lookup servers and at VL2 agents' cache, an update can lead to inconsistency. To resolve inconsistency, the cache-update protocol leverages a key observation: a stale host mapping needs to be corrected only when that mapping is used to deliver traffic. Specifically, when a stale mapping is used, some packets arrive at a stale LA—a ToR which does not host the destination server anymore. The ToR forwards such non-deliverable packets to a lookup server, triggering the lookup server to correct the stale mapping in the source's cache via unicast.

Figure 6. VL2 Directory System Architecture.



5. EVALUATION

In this section, we evaluate VL2 using a prototype running on an 80-server testbed and 10 commodity switches (Figure 7). Our goals are first to show that VL2 can be built from components available today, and second, that our implementation meets the objectives described in Section 1.

The testbed is built using the Clos network topology of Figure 4, consisting of three intermediate switches, three aggregation switches, and 4 ToRs. The aggregation and intermediate switches have 24 10Gbps Ethernet ports, of which 6 ports are used on each aggregation switch and 3

Figure 7. VL2 testbed comprising 80 servers and 10 switches.

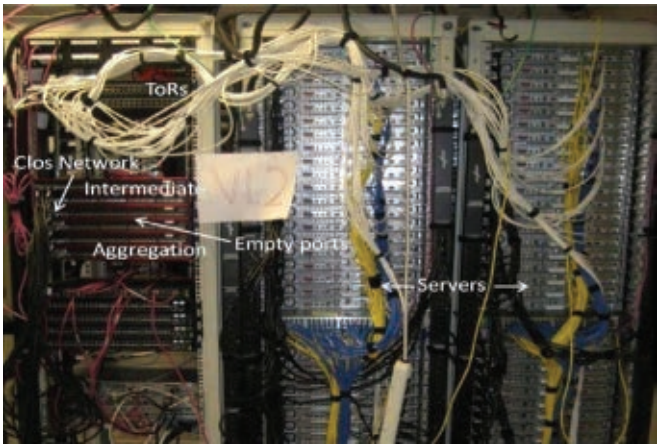
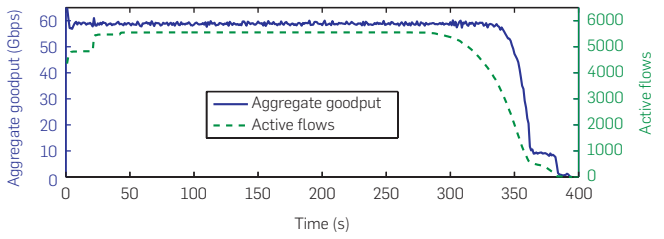


Figure 8. Aggregate goodput during a 2.7TB shuffle among 75 servers.



ports on each intermediate switch. The ToR switches have 2 10Gbps ports and 24 1Gbps ports. Each ToR is connected to 2 aggregation switches via 10Gbps links, and to 20 servers via 1Gbps links. Internally, the switches use commodity ASICs: Broadcom ASICs 56820 and 56514, although any switch that supports line rate L3 forwarding, OSPF, ECMP, and IPinIP decapsulation will work.

Overall, our evaluation shows that VL2 provides an effective substrate for a scalable data center network; VL2 achieves (1) 94% optimal network capacity, (2) a TCP fairness index of 0.995, (3) graceful degradation under failures with fast reconvergence, and (4) 50 K lookups/s under 10 ms for fast address resolution.

5.1. VL2 provides uniform high capacity

A central objective of VL2 is uniform high capacity between any two servers in the data center. How closely does the performance and efficiency of a VL2 network match that of a layer-2 switch with 1:1 oversubscription? To answer this question, we consider an all-to-all *data shuffle* stress test: all servers simultaneously initiate TCP transfers to all other servers. This data shuffle pattern arises in large-scale sorts, merges, and joint operations in the data center, for example, in Map/Reduce or DryadLINQ jobs.[7, 22] Application developers use these operations with caution today, because they are so network resource expensive. If data shuffles can be supported efficiently, it would have large impact on the overall algorithmic and data storage strategy.

We create an all-to-all data shuffle traffic matrix involving 75 servers. Each of 75 servers must deliver 500MB of data to each of the 74 other servers—a shuffle of 2.7TB from memory to memory. Figure 8 shows how the sum of the goodput over all flows varies with time during a typical run of the 2.7TB data shuffle. During the run, the sustained utilization of the core links in the Clos network is about 86%, and VL2 achieves an aggregate goodput of 58.8Gbps. The goodput is very evenly divided among the flows for most of the run, with a fairness index between the flows of 0.995[15] where 1.0 indicates perfect fairness (mean goodput per flow 11.4Mbps, standard deviation 0.75Mbps). This goodput is more than 10x what the network in our current data centers can achieve with the same investment.

We measure how close VL2 gets to the maximum achievable throughput in this environment by computing the goodput efficiency for this data transfer. Goodput efficiency is defined as the ratio of the sent goodput summed over all interfaces divided by the sum of the interface capacities. An efficiency of 1.0 would mean that all the capacity on all the interfaces is entirely used carrying useful bytes from the time the first flow starts to when the last flow ends. The VL2 network achieves an efficiency of 94%, with the difference from perfect being due to the encapsulation headers (3.8%), TCP congestion control dynamics, and TCP retransmissions.

This 94% efficiency combined with the fairness index of 0.995 demonstrates that VL2 can achieve uniform high bandwidth across all servers in the data center.

5.2. VL2 provides performance isolation

One of the primary objectives of VL2 is *agility*, which we define as the ability to assign any server, anywhere in the data center to any service (Section 1). Achieving agility critically depends on providing sufficient performance isolation between services so that if one service comes under attack or a bug causes it to spray packets, it does not adversely impact the performance of other services.

Performance isolation in VL2 rests on the mathematics of VLB—that any traffic matrix that obeys the hose model is routed by splitting to intermediate nodes in equal ratios (through randomization) to prevent any persistent hot spots. Rather than have VL2 perform admission control or rate shaping to ensure the traffic offered to the network conforms to the hose model, we instead rely on TCP to ensure that each flow offered to the network is rate limited to its fair share of its bottleneck.

A key question we need to validate for performance isolation is whether TCP reacts sufficiently quickly to control the offered rate of flows within services. TCP works with packets and adjusts their sending rate at the time scale of RTTs. Conformance to the hose model, however, requires instantaneous feedback to avoid oversubscription of traffic ingress/egress bounds. Our next set of experiments shows that TCP is "fast enough" to enforce the hose model for traffic in each service so as to provide the desired performance isolation across services.

In this experiment, we add two services to the network. The first service has a steady network workload, while the workload of the second service ramps up and down. Both the services' servers are intermingled among the 4 ToRs, so their traffic mixes at every level of the network. Figure 9 shows the aggregate goodput of both services as a function of time. As seen in the figure, there is no perceptible change to the aggregate goodput of service one as the flows in service two start up or complete, demonstrating performance isolation when the traffic consists of large long-lived flows. In Figure 10, we perform a similar experiment, but service two sends bursts of small TCP connections, each burst containing progressively more connections. These two experiments demonstrate TCP's enforcement of the hose model sufficient to provide performance isolation across services at timescales greater than a few RTT (i.e., 1–10ms in data centers).

5.3. VL2 directory system performance

Finally, we evaluate the performance of the VL2 directory system which provides the equivalent semantics of ARP in layer 2. We perform this evaluation through macro- and micro-benchmark experiments on the directory system. We run our prototype on up to 50 machines: 3–5 RSM nodes, 3–7 directory server nodes, and the remaining nodes emulating multiple instances of VL2 agents generating lookups and updates.

Our evaluation supports four main conclusions. First, the directory system provides high throughput and fast response time for lookups: three directory servers can handle 50K lookup/second with latency under 10ms (99th percentile latency). Second, the directory system can handle updates at rates significantly higher than the expected churn rate in typical environments: three directory servers can handle 12K updates/s within 600ms (99th percentile latency). Third, our system is incrementally scalable: each directory server increases the processing rate by about 17K for lookups and 4K for updates. Finally, the directory system is robust to component (directory or RSM servers) failures and offers high availability under network churn.

To understand the incremental scalability of the directory system, we measured the maximum lookup rates (ensuring sub-10ms latency for 99% requests) with 3, 5, and 7 directory servers. The result confirmed that the maximum lookup rates increases linearly with the number of directory servers (with each server offering a capacity of 17K lookups/s). Based on this result, we estimate the worst case number of directory servers needed for a 100K server data center. Using the concurrent flow measurements (Figure 3), we use the median of 10 correspondents per server in a 100s window. In the worst case, all 100K servers may perform 10 simultaneous lookups at the same time resulting in a million simultaneous lookups per second. As noted above, each directory server can handle about 17K lookups/s under 10ms at the 99th percentile. Therefore, handling this worst case will require a directory system of about 60 servers (0.06% of the entire servers).

Figure 9: Aggregate goodput of two services with servers intermingled on the ToRs. Service one's goodput is unaffected as service two ramps traffic up and down.

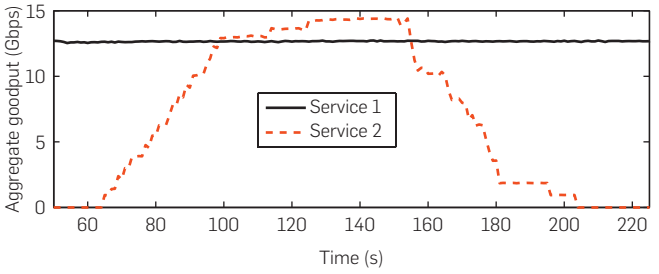
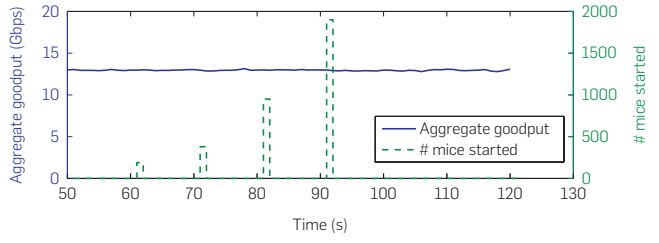


Figure 10. Aggregate goodput of service one as service two creates bursts containing successively more short TCP connections.



6. DISCUSSION

In this section, we address several remaining concerns about the VL2 architecture, including whether other traffic engineering mechanisms might be better suited to the DC than VLB, and the cost of a VL2 network.

Optimality of VLB: As noted in Section 4.2.2, VLB uses randomization to cope with volatility, potentially sacrificing some performance for a best-case traffic pattern by turning all traffic patterns (including both best-case and worst-case) into the average case. This performance loss will manifest itself as the utilization of some links being higher than they would under a more optimal traffic engineering system. To quantify the increase in link utilization VLB will suffer, we compare VLB's maximum link utilization with that achieved by other routing strategies on the VL2 topology for a full day's traffic matrices (TMs) (at 5 min intervals) from the data center traffic data reported in Section 3.

We first compare to *adaptive routing*, which routes each TM separately so as to minimize the maximum link utilization for that TM—essentially upper-bounding the best performance that real-time adaptive traffic engineering could achieve. Second, we compare to *best oblivious routing* over all TMs so as to minimize the maximum link utilization. (Note that VLB is just one among many oblivious routing strategies.) For adaptive and best oblivious routing, the routings are computed using respective linear programs in `cplex`. The overall utilization for a link in all schemes is computed as the maximum utilization over all routed TMs.

In Figure 11, we plot the CDF for link utilizations for the three schemes. We normalized the link utilization numbers so that the maximum utilization on any link for

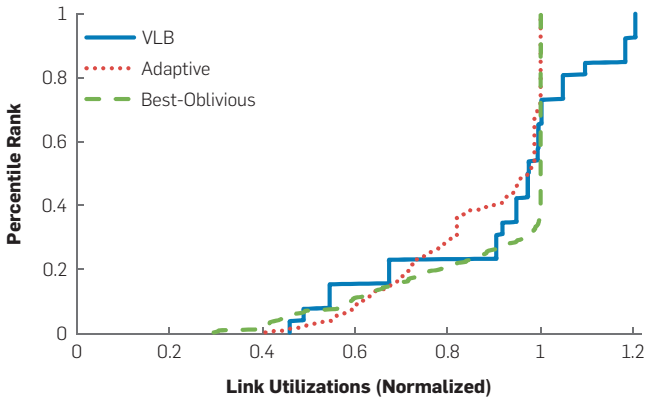
adaptive routing is 1.0. The results show that for most links, VLB performs about the same as the other two schemes. For the most heavily loaded link in each scheme, VLB's link capacity usage is at worst 20% higher than that of the other two schemes. Thus, evaluations on actual data center workloads show that the simplicity and universality of VLB costs relatively little capacity when compared to much more complex traffic engineering schemes. Moreover, adaptive routing schemes might be difficult to implement in the data center. Since even the "elephant" flows are about 100MB (see Figure 2), lasting about 1 s on a server with a 1Gbps NIC, any reactive traffic engineering scheme will need to run at least as frequently if it wants to react to individual flows.

Cost and Scale: With the range of low-cost commodity devices currently available, the VL2 topology can scale to create networks with no oversubscription between all the servers of even the largest data centers. For example, switches with 144 ports (D = 144) are available today for $150K, enabling a network that connects 100K servers using the topology in Figure 4 and up to 200K servers using a slight variation. Using switches with D = 24 ports (which are available today for $8K each), we can connect about 3K servers. Comparing the cost of a VL2 network for 35K servers with a conventional one found in one of our data centers shows that a VL2 network with no oversubscription can be built for the same cost as the current network that has 1:240 oversubscription. Building a conventional network with no oversubscription would cost roughly 14× the cost of a equivalent VL2 network with no oversubscription.

7. RELATED WORK

Data center network designs: There is great interest in building data center networks using commodity switches and a Clos topology.[2, 11, 20, 21] The designs differ in whether they provide layer-2 semantics, their traffic engineering strategy, the maturity of their control planes, and their compatibility with existing switches. Other approaches use the servers themselves for switching data packets.[1, 12, 13] VL2 also leverages the programmability of servers; however, it uses servers only to control the way traffic is routed as switch ASICs forward packets at less cost in power and dollars per Mbps.

Figure 11. CDF of normalized link utilizations for VLB, adaptive, and best oblivious routing schemes, showing that VLB (and best oblivious routing) come close to matching the link utilization performance of adaptive routing.



Valiant load balancing: Valiant introduced VLB as a randomized scheme for communication among parallel processors interconnected in a hypercube topology.[6] Among its recent applications, VLB has been used inside the switching fabric of a packet switch.[3] VLB has also been proposed, with modifications and generalizations,[18, 23] for oblivious routing of variable traffic on the Internet under the hose traffic model.[8]

Scalable routing: The Locator/ID Separation Protocol[9] proposes "map-and-encap" as a key principle to achieve scalability and mobility in Internet routing. VL2's control plane takes a similar approach (i.e., demand-driven host-information resolution and caching) but adapted to the data center environment and implemented on end hosts. SEATTLE[17] proposes a distributed host-information resolution system running on switches to enhance Ethernet's scalability.

Commercial networks: Data Center Ethernet (DCE)[4] by Cisco and other switch manufacturers shares VL2's goal of increasing network capacity through multipath. These industry efforts are primarily focused on consolidation of IP and storage area network (SAN) traffic, and there are few SANs in cloud-service data centers. Due to the requirement to support loss-less traffic, their switches need much bigger buffers (tens of MBs) than commodity Ethernet switches do (tens of KBs), hence driving their cost higher.

8. SUMMARY

VL2 is a new network architecture that puts an end to the need for oversubscription in the data center network, a result that would be prohibitively expensive with the existing architecture.

VL2 benefits the cloud service programmer. Today, programmers have to be aware of network bandwidth constraints and constrain server-to-server communications accordingly. VL2 instead provides programmers the simpler abstraction that all servers assigned to them are plugged into a single layer-2 switch, with hot spot–free performance regardless of where the servers are actually connected in the topology. VL2 also benefits the data center operator as today's bandwidth and control plane constraints fragment the server pool, leaving servers (which account for the lion's share of data center cost) underutilized even while demand elsewhere in the data center is unmet. Instead, VL2 enables agility: any service can be assigned to any server, while the network maintains uniform high bandwidth and performance isolation between services.

VL2 is a simple design that can be realized today with available networking technologies, and without changes to switch control and data plane capabilities. The key enablers are an addition to the end-system networking stack, through well-established and public APIs, and a flat addressing scheme, supported by a directory service.

VL2 is efficient. Our working prototype, built using commodity switches, approaches in practice the high level

of performance that the theory predicts. Experiments with two data center services showed that churn (e.g., dynamic reprovisioning of servers, change of link capacity, and microbursts of flows) has little impact on TCP goodput. VL2's implementation of VLB splits flows evenly and VL2 achieves high TCP fairness. On all-to-all data shuffle communications, the prototype achieves an efficiency of 94% with a TCP fairness index of 0.995.

Acknowledgments

Insightful comments from David Andersen, Jon Crowcroft, and the anonymous reviewers greatly improved the final version of this paper.

References

1. Abu-Libdeh, H., Costa, P., Rowstron, A., O'Shea, G., Donnelly, A. Symbiotic routing in future data centers. In *SIGCOMM* (2010).
2. Al-Fares, M., Loukissas, A., Vahdat, A. A scalable, commodity data center network architecture. In *SIGCOMM* (2008).
3. Chang, C., Lee, D., Jou, Y. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *IEEE HPSR* (2001).
4. Cisco. Data center Ethernet. http://www.cisco.com/go/dce.
5. Cisco: Data center: Load balancing data center services, 2004.
6. Dally, W.J., Towles, B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.
7. Dean, J., Ghemawat, S. MapReduce: simplified data processing on large clusters. In *OSDI* (2004).
8. Duffield, N.G., Goyal, P., Greenberg, A.G., Mishra, P.P., Ramakrishnan, K.K., van der Merwe, J.E. A flexible model for resource management in virtual private network. In *SIGCOMM* (1999).
9. Farinacci, D., Fuller, V., Oran, D., Meyer, D., Brimm, S. Locator/ID Separation Protocol (LISP). Internet-draft, Dec. 2008.
10. Greenberg, A., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D., Patel, P., Sengupta, S. Vl2: A scalable and flexible data center network. In *SIGCOMM* (2009).
11. Greenberg, A., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S. Towards a next generation data center architecture: Scalability and commoditization. In *PRESTO Workshop at SIGCOMM* (2008).
12. Guo, C., Wu, H., Tan, K., Shiy, L., Zhang, Y., Lu, S. DCell: a scalable and fault-tolerant network structure for data centers. In *SIGCOMM* (2008).
13. Guo, C., Wu, H., Tan, K., Shiy, L., Zhang, Y., Lu, S. BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM* (2009).
14. Hamilton, J. Cems: Low-cost, low-power servers for internet-scale services. In *Conference on Innovative Data Systems Research* (Jan 2009).
15. Jain, R. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, Inc., 1991.
16. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R. The nature of datacenter traffic: Measurements and analysis. In *IMC* (2009).
17. Kim, C., Caesar, M., Rexford, J. Floodless in SEATTLE: a scalable ethernet architecture for large enterprises. In *SIGCOMM* (2008).
18. Kodialam, M., Lakshman, T.V., Sengupta, S. Efficient and robust routing of highly variable traffic. In *HotNets* (2004).
19. Lamport, L. The part-time parliament. *ACM Trans. Comput. Syst. 16* (1998), 133–169.
20. Mudigonda, J., Yalagandula, P., Al-Fares, M., Mogul, J.C. Spain: Cots data-center ethernet for multipathing over arbitrary topologies. In *NSDI* (2010).
21. Touch, J., Perlman, R. Transparent interconnection of lots of links (TRILL): Problem and applicability statement. IETF RFC 5556 (2009).
22. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In *OSDI* (2008).
23. Zhang-Shen, R., McKeown, N. Designing a Predictable Internet Backbone Network. In *HotNets* (2004).

Albert Greenberg, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, Sudipta Sengupta,
Microsoft Research

James R. Hamilton,
Amazon Web Services

Barry University

Assistant Professor of Computer Science

The Department of Mathematics and Computer Science invites applications for a continuing contract faculty position in Computer Science with the rank of Assistant Professor, starting in Fall 2011.

Strong teaching skills, a commitment to research, and service to the Department and the University are expected. The Department has undergraduate majors in Mathematical Sciences, Computer Science and Computer Information Science. It also has a pre-Engineering program. The Department offers courses for the majors and service courses for all other Schools.

Qualifications:
The search is open to all areas of Computer Science, with a particular emphasis on candidates with research interests in software engineering, computer security or mobile computing.

A Ph.D. in Computer Science or closely related field is required.

Review of applications will start 02/14/2011 and will continue until the position is filled.

Interested and qualified candidates should send the following:

Complete curriculum vitae

Transcripts and Three letters of reference to:

CS Faculty Search Committee,
Dept. of MATH and CS, Barry University,
11300 NE 2nd Ave.,
Miami Shores, FL 33161.
Contact us by Fax (305)899-3610
By email to: mathcs@mail.barry.edu.

Barry University is a Catholic institution grounded in the liberal arts tradition and is committed to an inclusive community, social justice, and collaborative service.

Barry University is an Equal Employment Opportunity Employer.

Barry University does not discriminate applicants or employees for terms of employment on the basis of race, color, sex, religion, national origin, disability, veteran status, political affiliation or any other terms prohibited under the county ordinance, state or federal law.

NEC Laboratories America, Inc.

Research Staff Member - Machine Learning & Computer Vision

NEC Laboratories America, Inc. is a vibrant industrial research center, conducting research in support of NEC's U.S. and global businesses. Our research program covers many areas, reflecting the breadth of NEC business, and maintains a balanced mix of fundamental and applied research.

The Media Analytics Department in Cupertino, CA, is seeking an outstanding and enthusiastic researcher, with background in machine learning and computer vision, to work on developing visual recognition technologies for novel mobile applications, web services, and HCI solutions. We expect the candidates to be strong in conducting cutting edge research, and also passionate about turning research into high impact products and services. We encourage researchers to establish leadership in the research community, and maintain active research collaborations with top universities in the US.

Required Skills or Experience:
- PhD in Computer Science (or equivalent)
- Strong publication in top machine learning or computer vision conferences & journals
- Solid knowledge in math, optimization, and statistical inference
- Hands-on experiences in implementing large-scale learning algorithms and systems
- Great problem solving skills, with a strong desire for quality and engineering excellence
- Expert knowledge developing and debugging in C/C++

Desired Skills a Plus:
- Good knowledge developing and debugging on Linux
- Good knowledge developing in Java
- Experience with scripting languages such as Python, PHP, Perl, and shell scripts
- Experience with parallel/distributed computing
- Experience with algorithm implementation on GPU
- Experience with mobile or embedded systems
- Experience with image classification, object recognition, and visual scene parsing
- Ability to work on other media data, like textual and audio data

For consideration please submit your resume and a one-page research statement at http://www.nec-labs.com/careers/index.php.

EOE/AA/MFDV

NEC Laboratories America, Inc.

Research Staff Member - Energy Management

NEC Laboratories America, Inc.'s research program covers many areas, reflecting the breadth of NEC business, and maintains a balanced mix of fundamental and applied research. We focus on topics with strong innovations in the U.S. and place emphasis on developing deep competence in selective areas that are important to NEC business and which are ripe for technical breakthrough.

The **Energy Management Department** in Cupertino, CA, is seeking an outstanding and enthusiastic researcher with background in energy systems modeling and optimization to work on design of energy micro-grids. Candidates are expected to be strong in conducting cutting edge research, and also passionate about leading research threads and turning research into high impact products and services. We encourage researchers to establish leadership in the research community, and maintain active research collaborations with top universities in the US.

Required Skills or Experience:
- PhD in ME/EE(Power Systems)/CS/OR (or equivalent)
- Solid knowledge in math, optimization, and statistical analysis
- Hands-on experiences in implementing energy system models
- Great problem solving skills, with a strong desire for quality and engineering excellence
- Expert knowledge of optimization theory and tools
- Working knowledge of power and energy systems

Desired Skills:
- Knowledge of thermodynamic principles
- Knowledge of GAMS or equivalent optimization tools
- Experience in power sector

For consideration, please visit our career center at http://www.nec-labs.com/careers/index.php to submit your resume and a research statement.

EOE/AA/MFDV

Reykjavik University

School of Computer Science

Faculty position in computer systems

The School of Computer Science at Reykjavik University seeks to hire a faculty member in the field of computer systems. We are looking for a highly-qualified academic who, apart from developing her/his research programme, is interested in working with existing faculty, and in bridging between research, in one or more of the research areas within the School, in particular artificial intelligence, software engineering and theoretical computer science.

The level of the position can range from assistant professor to full professor, depending on the qualifications of the applicant. For information on the position, how to apply and the School of Computer Science at Reykjavik University, see http://www.ru.is/faculty/luca/compsysjob.html

University of Detroit Mercy
Computer Science/Software Engineering Tenure Track Faculty

The University of Detroit Mercy is currently seeking qualified candidates for the position of Assistant Professor beginning in August 2011. Please visit http://engsci.udmercy.edu/opportunities/index.htm for details and qualifications.

University of Wisconsin-Superior
Assistant Professor

Faculty Position, Computer Science, UW-Superior. Requires Doctorate in Computer Science, Electrical Engineering, Discrete Mathematics, or closely related field. See complete ad at http://www.uwsuper.edu/hr/employment/index.cfm CBC required. AA/EOE

Utah State University
Assistant Professor

Applications are invited for a faculty position at the Assistant Professor level, for employment beginning Fall 2011. Applicants must have completed a PhD in computer science by the time of appointment. The position requires demonstrated research success, a significant potential for attracting external research funding, excellence in teaching both undergraduate and graduate courses, the ability to supervise student research, and excellent communication skills.

USU offers competitive salaries and outstanding medical, retirement, and professional benefits (see http://www.usu.edu/hr/ for details). The department currently has approximately 280 undergraduate majors, 80 MS students and 27 PhD students. There are 17 full time faculty. The BS degree is ABET accredited. Utah State University is a Carnegie Research Doctoral extensive University of over 23,000 students, nestled in a mountain valley 80 miles north of Salt Lake City, Utah. Opportunities for a wide range of outdoor activities are plentiful. Housing costs are at or below national averages, and the area provides a supportive environment for families and a balanced personal and professional life. Women, minority, veteran and candidates with disabilities are encouraged to apply. USU is sensitive to the needs of dual-career couples. Utah State University is an affirmative action/equal opportunity employer, with a National Science Foundation ADVANCE Gender Equity program, committed to increasing diversity among students, faculty, and all participants in university life.

Applications must be submitted using USU's online job-opportunity system. To access this job opportunity directly and begin the application process, visit https://jobs.usu.edu/applicants/Central?quickFind=55484.

The review of the applications will begin on January 15, 2011 and continue until the position is filled. The salary will be competitive and depend on qualifications.

DEAN OF THE SCHOOL OF COMPUTER SCIENCE
FUDAN UNIVERSITY, SHANGHAI, CHINA

Founded 1905, Fudan University is one of the top five universities in China with over 26,000 undergraduate and graduate students, and over 3,600 international students earning more than 300 degrees through 28 schools and departments. In its quest to become recognized as a world class research university, Fudan has invested heavily on modern educational and research environments.

The newly established School of Computer Science is continuing a long tradition of computer science research at Fudan University which began in 1956 with the construction of China's first computer. The school is well positioned to remain at the forefront of international computing research and to continue to provide world-class education to its students. As Shanghai is rapidly growing into a world financial and business center, the School of Computer Science is strategically positioned to develop into a world-class center of excellence in computer science.

Learn more about Fudan University by visiting http://www.fudan.edu.cn.

Fudan University is seeking an innovative leader for the position of Dean of the School of Computer Science with academic and administrative leadership, and international vision.

Duties and Responsibilities
The Dean will be charged with the responsibilities of continuing to build the School towards achieving international stature. The successful candidate will do this as follows:
- Retain and recruit high caliber staff
- Continue to improve academic and research facilities
- Continue to improve curriculum
- Seek to strengthen strategic and cooperative relationships with internationally recognized universities

Qualification requirements
The successful candidate will demonstrate competency with a variety of experience:
- Leadership experience in a university department or research facility
- Carried out pioneered successful national and/or international research projects in computer science
- Published papers in leading international academic journals or international conference in recent six years
- No nationality restriction, Chinese as working language, fluent in English

Term and Salary
- 5 years each term
- Full-time assignment
- Compensation package negotiated during interview

Rights during the term
- The Dean will lead the school steering committee to operate and manage the financial and human resources based upon the guidelines and policies delegated by the university

To Apply:
Application timeline: From March 1 to March 31, 2011
Submit resume, degree certificates, certification of current employment, list of published papers during the last six years, 3-5 references and strategic vision and goal for the position.

Contact Information:
Fudan University, Office of Personnel
GE Qinghua , FENG Tao
Tel: 86-21-65642953 65654795 55664593
Email: yj@fudan.ac.cn

Fudan University School of Computer Science
YAO Xiaozhi
Tel: 86-21-51355555 ext. 28, Fax: 86-21-51355558
Email: xzyao@fudan.edu.cn
Website: http://www.cs.fudan.edu.cn

ACM TechNews Goes Mobile

iPhone & iPad Apps Now Available in the iTunes Store

ACM TechNews—ACM's popular thrice-weekly news briefing service—is now available as an easy to use mobile apps downloadable from the Apple iTunes Store.

These new apps allow nearly 100,000 ACM members to keep current with news, trends, and timely information impacting the global IT and Computing communities each day.

TechNews mobile app users will enjoy:

- ***Latest News:*** Concise summaries of the most relevant news impacting the computing world
- ***Original Sources:*** Links to the full-length articles published in over 3,000 news sources
- ***Archive access:*** Access to the complete archive of TechNews issues dating back to the first issue published in December 1999
- ***Article Sharing:*** The ability to share news with friends and colleagues via email, text messaging, and popular social networking sites
- ***Touch Screen Navigation:*** Find news articles quickly and easily with a streamlined, fingertip scroll bar
- ***Search:*** Simple search the entire TechNews archive by keyword, author, or title
- ***Save:*** One-click saving of latest news or archived summaries in a personal binder for easy access
- ***Automatic Updates:*** By entering and saving your ACM Web Account login information, the apps will automatically update with the latest issues of TechNews published every Monday, Wednesday, and Friday

The Apps are freely available to download from the Apple iTunes Store, but users must be registered individual members of ACM with valid Web Accounts to receive regularly updated content.

http://www.apple.com/iphone/apps-for-iphone/ **http://www.apple.com/ipad/apps-for-ipad/**

ACM TechNews

DOI:10.1145/1897852.1897878

Peter Winkler

Puzzled
Solutions and Sources

Last month (February 2011, p. 112) we posted a trio of brainteasers, including one as yet unsolved, concerning partitions of Ms. Feldman's fifth-grade class. Here, we offer solutions to at least two of them. How did you do?

1. Monday, Tuesday.
Solution. Recall that on Monday, Ms. Feldman partitioned her class into k subsets and on Tuesday repartitioned the same students into $k+1$ subsets. We were asked to show that at least two students were in smaller subsets on Tuesday than they were on Monday.

It turns out that a nice way to see this is to consider how much work each student contributed to his or her assigned project. Assume that all projects (both days) were equally demanding, with each requiring a total of one unit of effort. Assume, too, that the work was divided perfectly equitably, so a student in a subset of size m contributed $1/m$ units of effort. The total effort contributed on Monday was, of course, k and on Tuesday $k+1$, so some students must have contributed more of their effort on Tuesday than on Monday. But no individual student could have made up the full unit difference; the difference between $1/m$ and $1/n$ is less than 1 for any positive integers m and n. At least two students thus put in more effort on Tuesday and were therefore in smaller groups the second day.

This surprisingly tricky puzzle was brought to my attention by Ori Gurel-Gurevich of the University of British Columbia; it had appeared in the 1990 Australian Mathematics Olympiad.

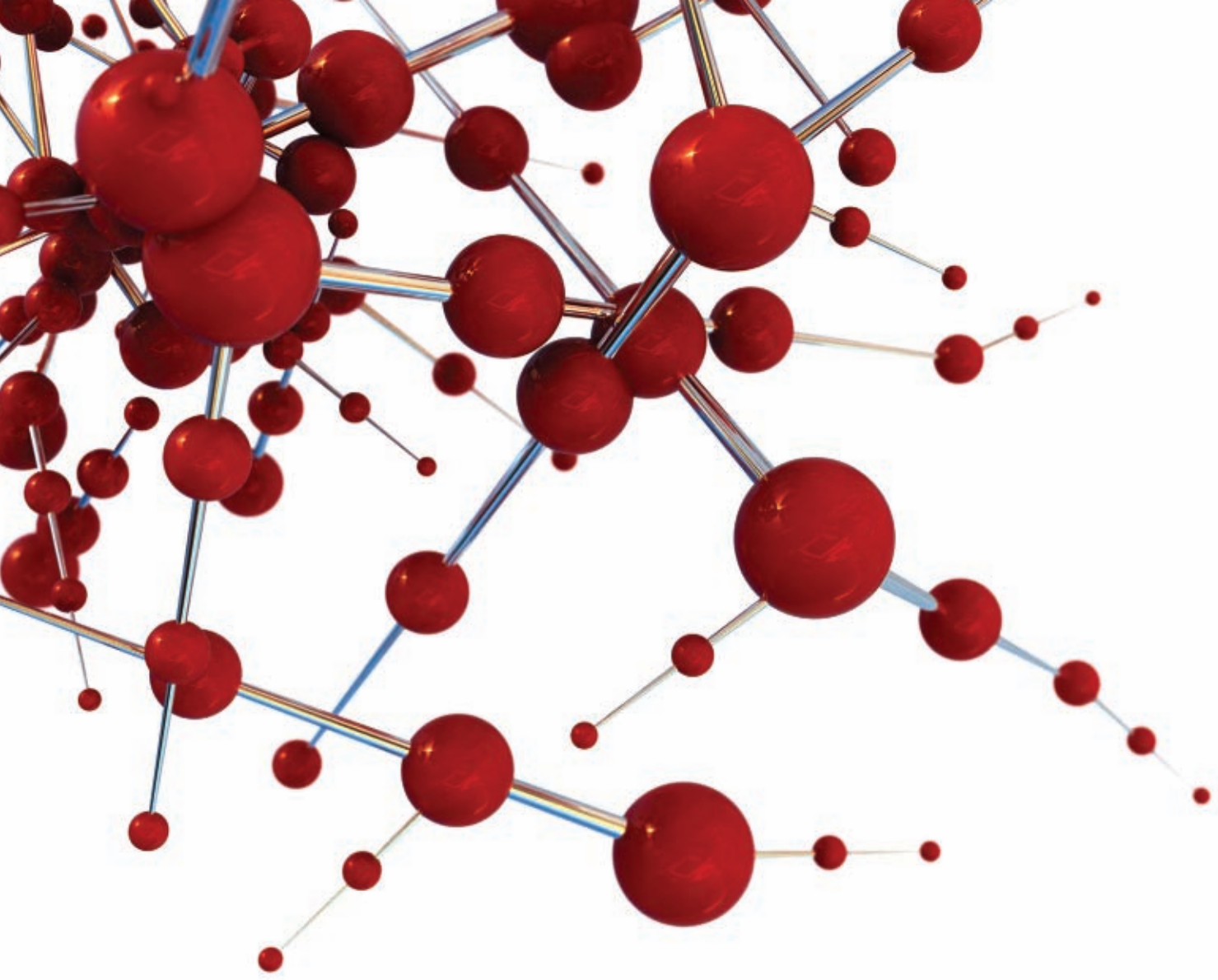
2. Unfriendly Partitions.
Solution. A partition of the kind Ms. Feldman wants, into two subsets, such that no student has more than half his/her friends in that student's own group, is called an "unfriendly partition." To see that an unfriendly partition exists, consider, for any partition, the number of broken friendships; that is, the number of pairs of friendly students who have been separated. Now choose a partition that maximizes this number; it must be unfriendly. Why? Because if student X has more friends in his/her subset than in the other subset, moving X from one to the other subset would yield a partition with more broken friendships.

3. Countably Infinite Graphs.
Unsolved. On Friday, when the class suddenly has a countably infinite number of students, Ms. Feldman can no longer apply these arguments to show that an unfriendly partition exists. The difficulty is that now there may be no partition with the maximum number of broken friendships. Moreover, even if there is a partition that breaks *infinitely* many friendships, the argument fails because switching student X as in Solution 2 doesn't give a contradiction.

Amazingly, no substitute argument has been found, nor has anyone come up with an example where no unfriendly partition exists. For (much) more information, see the marvelous article by Saharon Shelah and the late Eric Milnor, "Graphs with No Unfriendly Partitions," in *A Tribute to Paul Erdős*, A. Baker, B. Bollobás, and A. Hajnal, Eds., Cambridge University Press, 1990, 373–384. Shelah and Milnor constructed an "uncountable" graph with no unfriendly partitions; they also showed that every graph has an unfriendly partition divided into three subsets. The case of countably infinite graphs, when seeking to divide an unfriendly partition into two subsets, remains tantalizingly open—until, perhaps, you solve it.

Peter Winkler (puzzled@cacm.acm.org) is Professor of Mathematics and of Computer Science and Albert Bradley Third Century Professor in the Sciences at Dartmouth College, Hanover, NH.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.



[CONTINUED FROM P. 112] replicators ("rabbits" clone until they fill all memory), worms (traveling through networked computer systems, laying eggs), and plenty more.

Viruses were not a legacy I sought. Inevitably, someone would invent them; the idea requires only a simple biological analogy. But once it would escape into the general culture, there would be no way back, and I didn't want to make my professional life around it, lucrative as it might be. The manufacturers of spray-paint cans likely feel the same way...

Consider that our cities will get smart and be able to track us with cameras on the street and with microwaves that read the chips in our phones, computers, even embedded beneath our skin. The first commercial use will likely be to feed advertising to us, as in the 2002 Steven Spielberg film *Minority Report*. We'll inevitably live in an arms race against intrusive eyes, much as we guard against computer viruses today.

Stuxnet, the virus known to have invaded Iran's nuclear facilities, is apparently the first malicious code deliberately designed to disrupt targeted industrial processes, mutating on a schedule to avoid erasure, interrogating the computers it invades, and sending data back to its inventors. Stuxnet is able to reprogram Siemens-manufactured programmable logic controllers and hide the changes it introduces into them. Commands in Stuxnet code increase the frequency of rotors in centrifuges at Iran's Natanz uranium-enrichment plant so they fly apart. Yet much of Stuxnet's code is unremarkable, standard stuff, lacking advanced cloaking techniques.

Still, it's a wholly new thing—a smart virus with a grudge—evolving, self-aware, self-educating, craftily fulfilling its mission. Expect more to come. Countries hostile to the U.S. could likewise launch malware attacks against U.S. facilities, using Stuxnet-like code to attack the national power grid or other critical infrastructure.

Though seldom discussed, U.S. policy has traditionally been to lead in technology while selling last-generation tech to others. Thus we are able to defeat our own prior inventions, along with sometimes deliberately installed defects we might exploit later.

It's a wholly new thing—a smart virus with a grudge—evolving, self-aware, self-educating, craftily fulfilling its mission.

Stuxnet looks like a kluge with inventive parts. It does not hide its payload well or cover its tracks. It will not take much effort to greatly improve such methods (with, say, virtual machine-based obfuscation and novel techniques for anti-debugging), whatever the target. Once major players use them in nation-state rivalries, they will surely leak into commerce, where the stakes are immense for all of us. If Stuxtype, untraceable malware becomes a weapon of commerce, our increasingly global commercial competitiveness will take on a nastier edge.

Meanwhile, if living in space becomes routine, the related systems will demand levels of maintenance and control seldom required on Earth. Consider that the International Space Station spends most of its crew time just keeping the place running—and potentially can be corrupted with malware. So can many systems to come, as our environment becomes smarter and interacts with us invisibly, around the clock. Increasing interconnection of all systems will make smart sabotage a compelling temptation. So will malware that elicits data from our lives or corrupts systems we already have, in hopes we'll be compelled to replace them.

Now think beyond these early stages. What secondary effects could emerge? Seeds of mistrust and suspicion travel far. But that's the world we'll live in, with fresh problems we'll be able to attack but only if we've thought them through first. C

Gregory Benford (gbenford@uci.edu) is a professor of physics at the University of California, Irvine, and a novelist, including of *Timescape*, winner of the 1980 Nebula and British Science Fiction Awards.

Future Tense, one of the revolving features on this page, presents stories and essays from the intersection of computational science and technological speculation, their boundaries limited only by our ability to imagine what will and could be.

DOI:10.1145/1897852.1897879

Gregory Benford

Future Tense
Catch Me If You Can

Or how to lose a billion in your spare time...

I ENVISIONED AND wrote the first computer virus in 1969 but failed to see that viruses would become widespread. Technologies don't always evolve as we'd like. I learned this then but failed to catch the train I knew, even then, would soon leave the station. Further, I failed to see the levels of mistrust that would derive from malware generally. I also did not anticipate that seeds of mistrust could be blown by the gales of national rivalry through an Internet that would someday infiltrate every aspect of our lives.

At the Lawrence Radiation Laboratory I used the Advanced Research Projects Administration's network, or ARPANet, to send brief messages to colleagues in other labs running over the big, central computers we worshipped then. However, ARPANet email had a potentially pernicious problem—"bad code" that could arise when researchers sent something new (maybe accidentally), possibly sending yet other things awry.

One day I thought maybe I could add such code intentionally, making a program that would copy itself deliberately. The biological analogy was obvious; evolution would favor it, especially if designed to use clever methods to hide itself and tap other programs' energy (computing time) to further its own genetic ends.

So I wrote some simple code and sent it along in my next ARPANet transmission. Just a few lines in Fortran told the computer to attach them to programs being transmitted to a particular terminal. Soon it popped up in other programs and began propagating. By the next day it was in a lot of

otherwise unrelated code, so I wrote a memo, emphasizing to the mavens of the Main Computer that what I had done could likewise be done with considerably more malevolent intent. Moreover, viruses could move.

I avoided "credit" for the idea for a long time but gradually realized the virus-infection metaphor was inevitable, fairly obvious in fact. In the early 1970s it surfaced again at Livermore when a self-replicating program called Creeper infected ARPANet, just printing on a user's video screen "I'm the creeper, catch me if you can!" In response, users quickly wrote the first antivirus program, called Reaper, to erase Creeper. Various people reinvented the idea into the 1980s, when a virus called Elk Cloner infected early Apple computers. It was fixed quickly, but Microsoft software proved more vulnerable, and in 1986 a virus called Brain started booting up with Microsoft's disk operating system and spread through floppy disks, stimulating creation of the antivirus industry I had anticipated in 1970.

It is some solace, I suppose, that the 2010 second-best-selling virus-protection software was a neat little package called Vaccine. The same basic idea was adapted into a different kind of currency in the hands of renowned British biologist Richard Dawkins, coining the term "memes" to describe cultural notions that catch on and propagate through human cultural mechanisms. Ranging from pop songs we can't get out of our heads all the way up to the Catholic Church, memes express how cultural evolution occurs so quickly, as old memes give way to voracious new ones.

Nowadays there are nasty scrub-everything viruses of robust ability and myriad variations: Trojan horses, chameleons (acts friendly, turns nasty), software bombs (self-detonating agents, destroying without cloning themselves), logic bombs (go off given specific cues), time bombs (keyed by clock time), [CONTINUED ON P. 111]

ILLUSTRATION BY JOHN DAVID BIGL III

The 2012 ACM Conference on

Computer Supported Cooperative Work

February 11-15, 2012 | Seattle, Washington

Call for Submissions

CSCW is an international and interdisciplinary conference on technical and social aspects of communication, collaboration, and coordination. The conference addresses the design and use of technologies that affect groups, organizations, and communities. CSCW topics continue to expand as we increasingly use technologies to live, work and play with others.

This year we have adopted a new review process for papers and notes intended to increase their diversity and quality. The submission deadline is early to avoid conflicting with the CHI 2012 deadline. This enables us to include a revision cycle: Qualifying authors can revise their submissions after the initial reviews.

For complete details about CSCW's venues, please review the Call for Participation on our website www.cscw.2012.org. And please follow our progress at www.twitter.com/cscw2012 or by joining the CSCW 2012 Facebook group.

Conference Chairs: Steve Poltrock, Carla Simone

Papers and Notes Chairs: Gloria Mark, John Riedl, Jonathan Grudin

Also consider attending WSDM (wsdm2012.org) immediately before CSCW 2012.

Sponsored by

Association for Computing Machinery

SIGCHI

special interest group computer human interaction

Submission Deadlines

Papers & Notes
3 June 2011

Workshops
28 July 2011

Panels
Interactive Posters
Demonstrations
CSCW Horizon
Videos
Student Volunteers
9 September 2011

Doctoral Colloquium
16 October 2011

http://www.cscw2012.org