

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

02/2018 VOL.61 NO.02



The Next Phase in the Digital **REVOLUTION**

Dynamic Networks

**Teaching AI
and Humanity**

Going Serverless

Association for
Computing Machinery



Jason Jerald, PhD

The VR Book

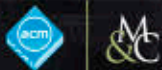
Human-Centered Design for Virtual Reality

The VR Book

*Human-Centered Design
for Virtual Reality*



Jason Jerald, Ph.D.



Dr. Jerald has recognized a great need in our community and filled it. The VR Book is a scholarly and comprehensive treatment of the user interface dynamics surrounding the development and application of virtual reality. I have made it required reading for my students and research colleagues. Well done!”

- Prof. Tom Furness, University of Washington, VR Pioneer



ISBN: 978-1-970001-12-9 DOI: 10.1145/2792790

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/vr>

Introducing *ACM Transactions on Human-Robot Interaction*

Now accepting submissions to ACM THRI

As of January 2018, the *Journal of Human-Robot Interaction* (JHRI) has become an ACM publication and has been rebranded as the *ACM Transactions on Human-Robot Interaction* (THRI).

Founded in 2012, the *Journal of HRI* has been serving as the premier peer-reviewed interdisciplinary journal in the field.

Since that time, the human-robot interaction field has experienced substantial growth. Research findings at the intersection of robotics, human-computer interaction, artificial intelligence, haptics, and natural language processing have been responsible for important discoveries and breakthrough technologies across many industries.

THRI now joins the ACM portfolio of highly respected journals. It will continue to be open access, fostering the widest possible readership of HRI research and information. All issues will be available in the ACM Digital Library.

Co-Editors-in-Chief Odest Chadwicke Jenkins of the University of Michigan and Selma Šabanović of Indiana University plan to expand the scope of the publication, adding a new section on mechanical HRI to the existing sections on computational, social/behavioral, and design-related scholarship in HRI.

The inaugural issue of the rebranded *ACM Transactions on Human-Robot Interaction* is planned for March 2018.

To submit, go to <https://mc.manuscriptcentral.com/thri>



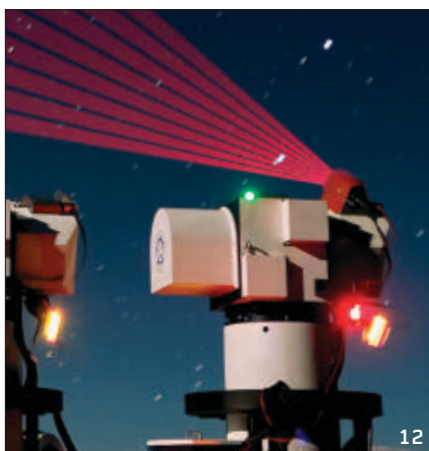
Departments

- 5 **From the Chair of ACM-W**
Achieving Gender Equity:
ACM-W Can't Do It Alone
By Jodi L. Tims
-
- 7 **Cerf's Up**
A Comprehensive
Self-Driving Car Test
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Toward an Equation that
Anticipates AI Risks
-
- 10 **BLOG@CACM**
Protecting the Power Grid, and
Finding Bias in Student Evaluations
John Arquilla considers the growth of cyberattacks on infrastructure, while Mark Guzdial wonders how beginning computer science students can possibly evaluate their teachers fairly.
-
- 31 **Calendar**
-
- 101 **Careers**

Last Byte

- 104 **Future Tense**
Welcome to the Singularity
Who can say no to the hive mind's promise of cybernetic immortality, for free?
By David Allen Batchelor

News

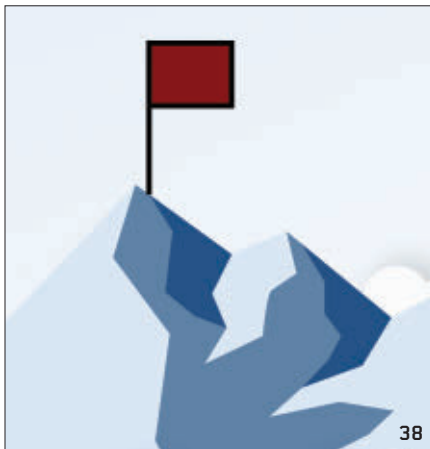


- 12 **Quantum Technology**
Forgoes Unconditional Security
to Extend Its Reach
Two projects in China demonstrate the possibility of global quantum key distribution networks.
By Chris Edwards
-
- 15 **Going Serverless**
Serverless computing lets businesses and application developers focus on the program they need to run, without worrying about the machine on which it runs, or the resources it requires.
By Neil Savage
-
- 17 **The War Over the Value**
of Personal Data
In a world increasingly dependent on turning personal data into profits, it is unclear how much that data is actually worth.
By Logan Kugler

Viewpoints

- 20 **Inside Risks**
Risks of Trusting
the Physics of Sensors
Protecting the Internet of Things with embedded security.
By Kevin Fu and Wenyan Xu
-
- 24 **Education**
The Inclusive and
Accessible Workplace
Maximizing the performance of neurodiverse talent.
By Sarah Wille and Daphne Sajoys-Brady
-
- 27 **Kode Vicious**
Reducing the Attack Surface
Sometimes you can give the monkey a less-dangerous club.
By George V. Neville-Neil
-
- 29 **Viewpoint**
Teaching Artificial Intelligence
and Humanity
Considering rapidly evolving human-machine interactions.
By Jennifer Keating and Illah Nourbakhsh
-
- 33 **Viewpoint**
Innovation from the Edges
How innovation originates from market participants with multiple perspectives about commercial value.
By Shane Greenstein

Practice



38

- 38 **Titus: Introducing Containers to the Netflix Cloud**
Approaching container adoption in an already cloud-native infrastructure.
By Andrew Leung, Andrew Spyker, and Tim Bozarth

- 46 **Research for Practice: Private Online Communication; Highlights in Systems Verification**
Expert-curated guides to the best of CS research.

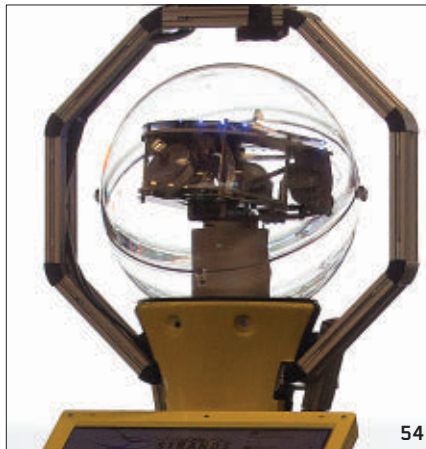
- 50 **Views from the Top**
Try to see things from a manager's perspective.
By Kate Matsudaira

Q Articles' development led by **acmqueue**
queue.acm.org

**About the Cover:**

The life and labor consequences of the digital revolution is just beginning to surface, say John Zysman and Martin Kenney, beginning on p. 54. Now is the time for citizens to rise up and take an active and vocal role in steering the path these *tools* will take. Cover illustration by Chris Whetzel.

Contributed Articles



54

- 54 **The Next Phase in the Digital Revolution: Intelligent Tools, Platforms, Growth, Employment**
Digital technology determines how (and even whether) people work as much as it determines how information produces economic activity.
By John Zysman and Martin Kenney



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/the-next-phase-in-the-digital-revolution>

- 64 **A Large-Scale Comparative Study of Beta Testers and Regular Users**
Beta testers should represent a future product's target users as much as possible.
By Vlasta Stavova, Lenka Dedkova, Martin Ukrop, and Vashek Matyas

Review Articles

- 72 **Elements of the Theory of Dynamic Networks**
The challenge of computing in a highly dynamic environment.
By Othon Michail and Paul G. Spirakis



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/elements-of-the-theory-of-dynamic-networks>

Research Highlights

- 83 **Technical Perspective**
Building Bug-Free Compilers
By Steve Zdancewic
- 84 **Practical Verification of Peephole Optimizations with Alive**
By Nuno P. Lopes, David Menendez, Santosh Nagarakatte, and John Regehr
- 92 **Technical Perspective**
Designing Algorithms and the Fairness Criteria They Should Satisfy
By Vincent Conitzer
- 93 **Which Is the Fairest (Rent Division) of Them All?**
By Kobi Gal, Ariel D. Procaccia, Moshe Mash, and Yair Zick



Association for Computing Machinery
Advancing Computing as a Science & Profession



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.

Acting Executive Director
Deputy Executive Director and COO
 Patricia Ryan
Director, Office of Information Systems
 Wayne Graves
Director, Office of Financial Services
 Darren Ramdin
Director, Office of SIG Services
 Donna Cappel
Director, Office of Publications
 Scott E. Delman

ACM COUNCIL

President
 Vicki L. Hanson
Vice-President
 Cherri M. Pancake
Secretary/Treasurer
 Elizabeth Churchill
Past President
 Alexander L. Wolf
Chair, SGB Board
 Jeanna Matthews
Co-Chairs, Publications Board
 Jack Davidson and Joseph Konstan
Members-at-Large
 Gabriele Anderst-Kotis; Susan Dumais;
 Elizabeth D. Mynatt; Pamela Samuelson;
 Eugene H. Spafford
SGB Council Representatives
 Paul Beame; Jenna Neefe Matthews;
 Barbara Boucher Owens

BOARD CHAIRS

Education Board
 Mehran Sahami and Jane Chu Prey
Practitioners Board
 Terry Coatta and Stephen Ibaraki

REGIONAL COUNCIL CHAIRS

ACM Europe Council
 Chris Hankin
ACM India Council
 Madhavan Mukund
ACM China Council
 Yunhao Liu

PUBLICATIONS BOARD

Co-Chairs
 Jack Davidson; Joseph Konstan
Board Members
 Phoebe Ayers; Anne Condon; Nikil Dutt;
 Roch Guerrin; Chris Hankin;
 Carol Hutchins; Yannis Ioannidis;
 Sue Moon; Michael L. Nelson;
 Sharon Oviatt; Eugene H. Spafford;
 Stephen N. Spencer; Alex Wade;
 Keith Webster; Julie R. Williamson

ACM U.S. Public Policy Office

Adam Eisgrau,
 Director of Global Policy and Public Affairs
 1701 Pennsylvania Ave NW, Suite 300,
 Washington, DC 20006 USA
 T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
 Deborah Seehorn,
 Interim Executive Director

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.

STAFF

DIRECTOR OF PUBLICATIONS

Scott E. Delman
 cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Lawrence M. Fisher

Web Editor

David Roman

Rights and Permissions

Deborah Cotton

Editorial Assistant

Jade Morris

Art Director

Andriy Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Bernadette Shade

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

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

CONTACT POINTS

Copyright permission
 permissions@hq.acm.org

Calendar items
 calendar@cacm.acm.org

Change of address
 acmhelp@acm.org

Letters to the Editor
 letters@cacm.acm.org

WEBSITE

<http://cacm.acm.org>

AUTHOR GUIDELINES

<http://cacm.acm.org/about-communications/author-center>

ACM ADVERTISING DEPARTMENT

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

Advertising Sales Account Manager

Ilia Rodriguez
 ilia.rodriguez@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

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

EDITORIAL BOARD

EDITOR-IN-CHIEF

Andrew A. Chien
 eic@cacm.acm.org

Deputy to the Editor-in-Chief

Lihan Chen
 cacm.deputy.to.eic@gmail.com

SENIOR EDITOR

Moshe Y. Vardi

NEWS

Co-Chairs

William Pulleyblank and Marc Snir

Board Members

Monica Divitini; Mei Kobayashi;
 Michael Mitzenmacher; Rajeev Rastogi;
 François Sillion

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;
 John Leslie King; Paul Rosenbloom

Board Members

Stefan Bechtold; Michael L. Best;
 Judith Bishop; Mark Guzdial;
 Richard Ladner; Carl Landwehr;
 Beng Chin Ooi; Loren Terveen;
 Marshall Van Alstyne; Jeannette Wing

PRACTICE

Chair

Stephen Bourne and Theo Schlossnagle

Board Members

Eric Allman; Samy Bahra; Peter Bailis;
 Terry Coatta; Stuart Feldman; Nicole Forsgren;
 Camille Fournier; Benjamin Fried;
 Pat Hanrahan; Tom Killalea; Tom Limoncelli;
 Kate Matsudaira; Marshall Kirk McKusick;
 Erik Meijer; George Neville-Nel;
 Jim Waldo; Meredith Whittaker

CONTRIBUTED ARTICLES

Co-Chairs

James Larus and Gail Murphy

Board Members

William Aiello; Robert Austin;
 Elisa Bertino; Kim Bruce; Alan Bundy;
 Peter Buneman; Carl Gutwin;
 Yannis Ioannidis; Gal A. Kaminka;
 Ashish Kapoor; Kristin Lauter; Igor Markov;
 Bernhard Nebel; Lionel M. Ni; Adrian Perrig;
 Marie-Christine Rousset; Krishan Sabnani;
 Ron Shamir; Alex Smola; Josep Torrellas;
 Sebastian Uchitel; Michael Vitale;
 Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Shriram Krishnamurthi

Board Members

Martin Abadi; Amr El Abbadi; Sanjeev Arora;
 Michael Backes; Maria-Florina Balcan;
 Andrei Broder; Doug Burger; Stuart K. Card;
 Jeff Chase; Jon Crowcroft; Alexei Efros;
 Alon Halevy; Sven Koenig; Steve Marschner;
 Tim Roughgarden; Guy Steele, Jr.;
 Margaret H. Wright; Nicolai Zeldovich;
 Andreas Zeller

WEB

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;
 Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2018 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

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 \$269.

ACM Media Advertising Policy

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

Single Copies

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

COMMUNICATIONS OF THE ACM

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

POSTMASTER

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

Printed in the USA.



Association for
Computing Machinery



DOI:10.1145/3173556

Jodi L. Tims

Achieving Gender Equity: ACM-W Can't Do It Alone

THE APRIL 2017 edition of *Communications* included an editorial from ACM-W on the status of gender diversity in computing and the painstakingly slow progress being made toward an equitable representation of women in our discipline. In that editorial, Valerie Barr highlighted ACM's commitment to diversity more generally via a new ACM Council on Diversity. Work on the establishment of this Council is continuing and ACM-W looks forward to being a part of this broader effort.

In the interim, our work on behalf of women in computing continues in earnest. Since the beginning of 2017, there have been 26 new ACM-W Student Chapters chartered. During the 2017–2018 academic year, a record 29 Celebrations of Women in Computing are being held in locations all over the world. Our connections to ACM SIGs and partner organizations outside of ACM are strengthening. We recently endorsed important legislation pending in the U.S. Congress that holds promise to increase computing education to girls in the elementary grades.^a With several other professional organizations in science and mathematics, ACM-W is participating in a three-year project to gather significant and currently unavailable global data about women's participation in our disciplines.^b

I could fill this column with more examples of the work of our many dedicated ACM-W volunteers, but I think it is important to focus a bit on

a nagging question that many of us who work so hard in this space of gender equity in computing have. Why, with so much sustained effort by so many individuals and organizations, is progress toward gender equity so slow?

Of course, if there was a known answer to this question we would not still be asking it. We know that ultimately there must be significant systemic change on many fronts including pre-tertiary education, workplace environment, and societal perception of computing professionals. Systemic change is difficult work that can take many years to realize. Systemic change will not be achieved if responsibility for realizing gender equity is viewed as belonging to the women in computing or the many organizations whose primary focus is gender equity. The change will occur only when every individual computing professional accepts responsibility for making it happen.

ACM as an organization impacts our profession through the individual and collective work of its membership. In order for ACM to have a bigger influence on the state of gender equity in computing, every ACM member, regardless of gender, must do her or his part to understand the problem, create inclusive environments, speak out on issues that impact the experience of women in computing, and advocate for social change that will turn the tide long-term. Individual investment in the work of gender diversity will transform the special interest groups, chapters, and conferences of ACM in ways that will redefine our external image and expand our ability to influence societal change.

So what can an individual do on a daily basis to ensure her/his environment fosters inclusiveness? I posed this question out to a few members of the ACM-W Council and received lots of good suggestions. Here are just a few of them:

- Once a month become familiar with at least one woman in your office or on your campus that you do not know and then introduce them to your peers.

- Read about unconscious bias and stereotype threat so that you can recognize these things when they occur and speak out against them.

- Ensure that original ideas are attributed to the first person (not the first male) to make the suggestion.

- Say a few sincere, kind words to a female colleague regarding her work.

There were many more great ideas generated and I am confident the ACM community has even more to offer. I invite you to read a recent blog post (<https://cacm.acm.org/blogs/blog-cacm/224005>) and contribute your own strategies for supporting women in computing in the workplace.

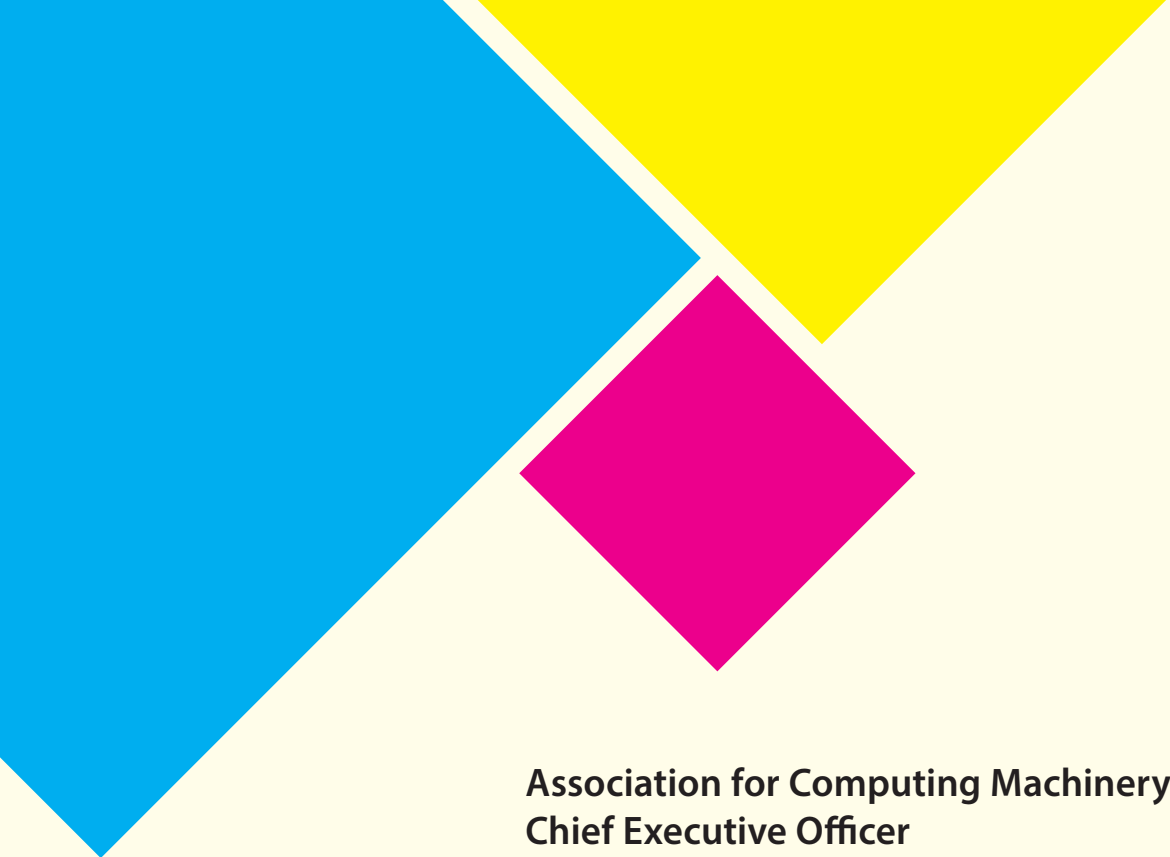
ACM has the potential to set the standard for what it means to be an organization committed to solving issues of gender diversity in computing. It may even be possible that a day will come when ACM-W will no longer need to exist. Until then, ACM-W will continue to engage in activities that support, celebrate, and advocate for women in computing and we welcome all who will join with us. 

Jodi L. Tims (jltims@bw.edu) is chair of the computer science department at Baldwin Wallace University, Berea, OH, USA, and chair of ACM-W.

Copyright held by author.

^a Press release of this legislation can be found at <http://bit.ly/2uNf1gA>.

^b The Gender Gap in Science project can be followed at <https://icsugendergapinscience.org/>.



Association for Computing Machinery (ACM) Chief Executive Officer

ACM, the Association for Computing Machinery, invites applications for the position of Chief Executive Officer (CEO).

ACM is the oldest and largest educational and scientific computing society with nearly 100,000 members worldwide. The association has an annual budget of \$75 million, 75 full-time staff in New York and Washington D.C., a rich publications program that includes 90 periodicals in computing and hundreds of conference proceedings, a dynamic set of special interest groups (SIGs) that run nearly 300 conferences/symposia/workshops each year, initiatives in India, China, and Europe, and educational and public policy initiatives. ACM is the world's premiere computing society.

The ACM CEO serves as the primary executive responsible for the formulation and implementation of ACM strategic direction, for representing ACM in the worldwide computing community, and for overall management of the affairs of the association. The successful candidate will have high professional standing in the computing field, executive experience, leadership skills, and a vision of the future of professional societies and computing. The CEO reports to the ACM President. It is not a requirement that the CEO work from ACM's New York headquarters, but must be able to travel frequently to headquarters and other ACM meetings. The full job description and details on how to apply can be found at: ceosearch.acm.org

The ACM is an equal opportunity employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, age, protected veteran status or status as an individual with disability.



**Association for
Computing Machinery**



Vinton G. Cerf

DOI:10.1145/3177753

A Comprehensive Self-Driving Car Test

Every few years, I have to pass a test from the Department of Motor Vehicles to drive my car in Virginia (and the rest of the U.S.). Shouldn't a self-driving car be required

to do the same thing? Actually, the Waymo self-driving car passes a more comprehensive set of tests than humans do, as I found out after asking about its safety report.^a Disclaimer: I work for Google, which is an Alphabet^b company and Waymo is a sister company.

What struck me as interesting about Waymo's approach to safety is the scope of the design and testing regime that informs the company's assessment of the vehicle's safety. For such vehicles to work at all, a raft of sensors is needed to provide the vehicle's software with situation awareness at all times. Unlike human drivers, the self-driving car can continuously sense its 360-degree surroundings using multiple sensors: color-aware visible light cameras, radar transceivers, three lidar^c transceivers (short, medium, and long range), audio detectors, and GPS receivers. Moreover, a great deal of redundancy is built into the system to provide back-up capacity in the event of various failure scenarios. The list is long: back-up braking, computing, steering power, collision detection and avoidance systems, and redundant inertial measurement systems.

The Waymo vehicles have accumulated four million miles of driving on city streets in California, Washington state, Arizona, and Texas. Each day, as many as 25,000 virtual Waymo self-

driving vehicles drive up to eight million miles in simulation for an accumulated total of 2.5 billion simulated miles during the course of self-driving car development.

From the safety report: "Waymo has set up a private, 91-acre, closed-course testing facility in California specially designed and built for our own unique testing needs. This private facility, nicknamed "Castle," is set up like a mock city, including everything from high-speed roads to suburban driveways to a railroad crossing. Our team uses this and other closed-course facilities to validate new software before it's released to our fleet of vehicles on the road, and also to stage challenging or rare scenarios so our vehicles gain experience with unusual situations. On our closed course, we're able to conduct thousands of "structured tests" that recreate specific scenarios for learning and testing. To power our simulator, we've developed more than 20,000 simulation scenarios at Castle. Each recreates a driving situation we want to practice—an aggressive driver barreling out of a driveway, or a pedestrian suddenly emerging from a parked car—that might take hundreds of thousands of driving miles to encounter on public roads. We've staged people jumping out of canvas bags or portable toilets on the side of the road, skateboarders lying on their boards, and thrown stacks of paper in front of our sensors. This "structured testing"

is key to accelerating the progress of our technology and ensuring safety of our vehicles in both everyday and challenging driving situations."

The simulation capability is particularly important since it allows Waymo to test any new software or hardware releases with large numbers of scenario variations in parallel that would take far too long to test in the real world. The real-world tests provide detailed sensor data, which can be recorded, played back in simulation, and artificially varied to create comprehensive situational testing.

While the U.S. Department of Transportation has recommended that self-driving vehicles should be able to demonstrate at least 28 core competencies adapted from research by California Partners for Advanced Transportation Technology (PATH) at the Institute of Transportation Studies at University of California, Berkeley, Waymo has identified a total of 47 core competencies and endless variations within them for validating safety. The cars can continuously test the condition of all onboard systems and have been designed to assume a minimal risk condition (NASA calls this "safe mode") if a hazardous situation develops.

Care has been given to assure the vehicles can detect and react properly to the presence of emergency vehicles and provision has been made to allow the cars to interact with law enforcement and other emergency response personnel as well as communicating with the passengers on board the self-driving cars. I don't know about you, but I am really impressed by Waymo's comprehensive focus on safety and the implications for reducing the hazards of human-driven cars! 

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

Editor's Note: See another side of the sensor story on p. 20.

a <https://waymo.com/safetyreport/>

b <https://abc.xyz/>

c "light radar"



Advertise with ACM!

Reach the innovators
and thought leaders
working at the
cutting edge
of computing
and information
technology through
ACM's magazines,
websites
and newsletters.



Request a media kit
with specifications
and pricing:

Ilia Rodriguez
+1 212-626-0686
acmm mediasales@acm.org



DOI:10.1145/3177403

Toward an Equation that Anticipates AI Risks

IN HIS “INSIDE RISKS” Viewpoint “The Real Risks of Artificial Intelligence” (Oct. 2017), David Lorge Parnas wrote that “artificial intelligence” remains undefined while highlighting his concern that AI could yet render humans superfluous and aid authoritarian regimes looking to centralize their hold on political power. He also said AI could yet produce untrustworthy potentially dangerous devices and systems.

Among the very human psychological factors driving human fear are being financially or medically dependent on others, the expectation of physical or mental pain, unintentionally hurting others (such as by causing a car crash), being irresponsible (such as by forgetting an infant left in a car on a hot day), or simply being embarrassed about some inappropriate social behavior. Many of us fear losing our privacy and jobs, thoughtlessly insulting colleagues, being overly controlled by governments and corporations, suffering injustice, or being victimized by violence, especially if avoidable. It is our darkest fears that actually protect us the most. Could AI intensify such fears to levels beyond what we already know?

History records numerous instances of humans delivering slavery, humiliation, and genocide through even the simplest of technologies. Consider that swords, rope, and horses have allowed a handful of leaders to control vast populations. Pirates armed with guns to target passengers on airplanes or cruise ships are another threat to life and property. Other non-computational technologies that could, at first glance, appear so unsophisticated as to seem harmless include poison gas (as used by the Nazis for mass murder) and knives (as used to commandeer commercial airplanes in the 9/11 terror attacks). Even the simplest devices can be the riskiest, representing a much greater threat than any undefined super AI.

Measuring the magnitude of risk for a new device or for an entire category of technology is not straightforward and becomes even more difficult in light of AI's incomplete scientific definition, which might even be self-serving. Whether human-like or self-aware, self-motivated machines are more harmful than the tools our Paleolithic ancestors might have used long ago is an open question. Regardless how difficult it is to measure AI's potential, computer scientists would benefit from developing a new equation to estimate that risk, before the technology itself would become widespread, embedded in the Internet of Things. Like Drake's Equation¹—created by astrophysicist Frank Drake to estimate the number of potentially communicative extraterrestrial civilizations in the Milky Way galaxy, an equation defining quantitatively the boundaries of machine intelligence and its potential risk²—would stimulate further scientific debate around AI and help define—scientifically—AI benefits and risks.

References

1. Drake, N. How my dad's equation sparked the search for extraterrestrial intelligence. *National Geographic* (June 30, 2014).
2. Dietterich, T.G. and Horvitz, E.J. Rise of concerns about AI: Reflections and directions. *Commun. ACM* 58, 10 (Oct. 2015), 38–40.

Uri Kartoun, Cambridge MA, USA

Author Responds

Although I mentioned that others, including famous people, had expressed the fear that AI could “render humans superfluous,” I do not share their view. As I explained, my concern is that programs designed by AI methods, rather than based on solid mathematical models, will be untrustworthy. I also said the term “artificial intelligence” has not been properly defined. Without a definition, no formula can reliably predict the risk of using it.

David Lorge Parnas,
Ottawa, Ontario, Canada

Final Knowledge with Certainty Is Unobtainable

Martin E. Hellman's Turing Lecture "Cybersecurity, Nuclear Security, Alan Turing, and Illogical Logic" (Dec. 2017) did not say the crypto wars from the 1970s have returned, thus threatening to overturn Hellman et al.'s own victory over mandatory government access to information in communication devices. Also not said was that the common understanding of mathematician Kurt Gödel's results have been revised by mathematicians and logicians because they were based on first-order logic, which is being replaced by higher-order logic in computer science with knock-on effects.

Return of the crypto wars and revision of the common understanding of Gödel's results illustrate that final knowledge with certainty is unobtainable in computer science, as it is in all other fields, and that further extensions, reinterpretations, and revisions are always possible through a process I would call "progressive knowing" that is never finished and never certain.¹

The crypto wars have resumed through a current proposal from government security contractors that aims to provide government access to all Internet of Things devices in a way that only the government would be able to use to exfiltrate information. A public key would be required in each new device sold in the U.S. such that when a packet arrives that decrypts using that public key, the decrypted packet would become the "bootloader" for a virtual machine to take over the device, even as it is being used. Corresponding private keys can be protected against a single point of failure by splitting them into multiple pieces and storing each piece in a different secure government facility. Government access could, over time, be enforced by requiring all new devices sold in the U.S. interactively verify they can be accessed by the government before they would be allowed to connect to the U.S. public Internet. A device from another country would be allowed to connect domestically only after arrangements were made over the Internet with a foreign security service.

Government access might be used pursuant only to a court order. But there is nothing in the physical arrangements of the proposal for mandatory government access to prevent government surveillance. Such access was also a principle objection to the original technically defective government proposal that Hellman et al. confronted in the 1970s. By correcting these technical defects, the new proposal threatens to overturn the victory in the earlier crypto wars.

Meanwhile, Gödel's results were based on first-order logic, but every moderately powerful first-order theory is inconsistent. Consequently, computer science is changing to use higher-order logic. However, logicians have shown there are proofs of theorems in higher-order logic that cannot be expressed through text alone, thus overturning a long-held nominally established philosophical dogma about mathematical theories—that all theorems of a theory can be computationally generated by starting with axioms and mechanically applying rules of inference.¹ "Inexpressibility" means it is mathematically provable that it will be forever necessary for computer science to invent new notations for mathematical proofs.

Reference

1. Hewitt, C. *Strong Types for Direct Logic*. HAL Archive; <https://hal.archives-ouvertes.fr/hal-01566393>

Carl Hewitt, Palo Alto, CA, USA

Author Responds

Hewitt is correct that the crypto wars have continued, but the victory I mentioned still holds: establishing that independent researchers could publish papers free from government interference. His comments on Gödel's results go beyond my mathematical knowledge but do not affect the main point I made about logic being just one way of knowing about the world, and an incomplete one at that.

Martin E. Hellman, Stanford, CA, USA

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

©2018 ACM 0001-0782/18/2

Coming Next Month in COMMUNICATIONS

A Programmable Programming Language

How Can We Trust a Robot?

The Evolution Toward Soft(er) Products

The Wisdom of Older Technology Users

Responsible Research with Crowds

Computational Social Science Equals Computer Science + Social Science

Bitcoin's Underlying Incentives

Operational Excellence in April Fool's Pranks

Monitoring in a DevOps World

Time-Inconsistent Planning

SSL Certification Reissues and Revocations in the Wake of Heartbleed

Q&A with Yann LeCun

Plus the latest news about virtual life, construction in the 21st century, and the state of faking.

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.



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

DOI:10.1145/3171576

<http://cacm.acm.org/blogs/blog-cacm>

Protecting the Power Grid, and Finding Bias in Student Evaluations

John Arquilla considers the growth of cyberattacks on infrastructure, while Mark Guzdial wonders how beginning computer science students can possibly evaluate their teachers fairly.



John Arquilla The Rise of Strategic Cyberwar?

<http://bit.ly/2htUUe5>

September 25, 2017

Over the past few years, a troubling hacking trend has emerged, characterized by serious intrusions into electric power infrastructures. Most of this activity has been system-mapping across several countries, ranging from the U.S. to Ireland, and on to Switzerland and Turkey. There is evidence of actual attacks, notably in Ukraine's Ivano-Frankivsk region in December 2015, when power was knocked out. The prime suspects in these intrusions appear to be Russia-friendly hacker groups known variously as "Dragonfly" and "Energetic Bear," among other names.

The attention to power grids seems to have emerged hand in hand with a growing hacker interest in the broader realm of automated system controls, commonly called SCADA (supervisory control and data acquisition), whose uses are increasing across the spectrum of activities essential to a modern society's

ability to function smoothly. This focus on mapping, and launching the occasional attack, on infrastructure may herald the coming rise of strategic cyberwarfare as a means of striking in costly, disruptive ways at an adversary without a need to defeat opposing military forces. Further, the possibility such attacks can be launched anonymously, or at least "deniably" via proxies, may reduce the risk of retaliatory conflict escalation.

Cyberwar seems to be following a path similar to that followed during the rise of air warfare a century ago, when military thinkers like the American Billy Mitchell and the Italian Giulio Douhet were holding forth with their views about the independent, war-winning potential of strategic attack from the air. Douhet went so far as to encourage the use of chemical weapons in aerial bombing of population centers, to hasten the psychological breaking-point he was sure would follow. While Douhet's call for chemical attack from the air was almost completely rejected worldwide, there was still broad acceptance of his no-

tion that civilian populations would not bear up under bombardment.

Strategic bombing campaigns from World War II to Korea, Vietnam, and beyond, have been repeatedly launched—with very few successes, per the study by Robert Pape, "Bombing to Win" (<http://bit.ly/2iU3zLH>). NATO's successful 78-day Kosovo air war in 1999 against Serbia may be the lone clear exception that proves the rule about how difficult it is to win by means of aerial bombardment.

"Shock and awe" from the air just does not work. On the other hand, the wars of the past 75-plus years have repeatedly seen the close air support of military and naval forces by attack aircraft fundamentally transform and dominate warfare on land and at sea.

What if cyberwar follows a similar path? Recent indicators of hacker interest in infrastructure may be a sign cyber attack is being viewed primarily in strategic terms—that is, as a way of inflicting material and psychological costs on the enemy—instead of as a means of improving the performance of forces in battle. In World War II, Germany and Japan first focused, respectively, on the tremendous combat value of close air support on land and carrier operations at sea. Their opponents were slow off the mark, and the outcome of the war hung in the balance for years.

If interest in mapping power infrastructures is a sign cyber is viewed as a form of strategic attack, it seems the same wrongheaded path that misled so many about which aspect of air power to emphasize is being pursued. If the widespread destruction of strategic aerial

bombardment has seldom worked, “mass disruption” from cyber attacks on infrastructure is even less likely to achieve the desired psychological effects. Such attacks will kindle great rage among those affected, leading to conflict escalation. In that larger conflict, the side that has learned to use cyber at the tactical level will prevail.

It may seem reassuring that the apparently Russia-friendly hacker groups are focusing on infrastructure targets, the implication being this suggests an emphasis on developing strategic, rather than tactical, cyberwar capabilities. But this is not an either-or situation. Aggressors might be cultivating battlefield cyber capabilities as well. How might one tell? One clue could be that infrastructure probes and attacks to date have generally not used zero-day exploits; almost all have been simple, employing watering-hole techniques (lying in wait at frequented sites), man-in-the-middle attacks (rerouting individuals’ Internet traffic), and other basic methods. The world’s cyber aggressors may have a whole other gear we have not seen, which will be revealed in a shooting war.

It is this latter sort of militarized conflict that David Ronfeldt and I envisioned when we wrote “Cyberwar Is Coming!” (<http://bit.ly/2AtTlbt>) a quarter-century ago. It is in its effects on the course of battles—on land, at sea, in the air, and outer space—that cyber will show its true potential to transform warfare in the 21st century.

Cyberwar is not simply a lineal descendant of strategic air power; rather, it is the next face of battle.



Mark Guzdial
Evaluating Computer Science Undergraduate Teaching: Why Student Evaluations Are Likely Biased

<http://bit.ly/2AwBT3H>

April 23, 2017

Our campus has been having discussions about student evaluations of teaching. Our Center for Teaching and Learning circulated a copy of an article by Carl Wieman from *Change* magazine, “A Better Way to Evaluate Undergraduate Teaching” (<http://bit.ly/2ipatVy>).

Wieman argues we need a better way to evaluate teaching; student

evaluations do not correlate with desirable outcomes (as described at <http://bit.ly/2iXrn17>) and are biased.

“To put this in more concrete terms, the data indicate that it would be nearly impossible for a physically unattractive female instructor teaching a large required introductory physics course to receive as high an evaluation as that of an attractive male instructor teaching a small fourth-year elective course for physics majors, regardless of how well either teaches.”

Wieman suggests a Teaching Practices Inventory (<http://bit.ly/2ioK5Le>) as a better way to evaluate undergraduate teaching. Using practices that are evidence-based is likely to lead to better outcomes. This hasn’t been an easy sell, as Wieman discovered at the White House Office of Science and Technology Policy (<http://bit.ly/2B1giUo>). It has not gone over well on my campus, either.

Scholars like Nira Hativa argue student evaluations are an effective way to recognize good teaching (see <http://amzn.to/2ingr94>). Student evaluation of teaching is easy, and is current standard practice, which is difficult to change. Wieman’s Teaching Practices Inventory has been called “radical” on my campus.

I am not a scholar of studies about student evaluation of teaching. I study computing education. From what I know about computer science and unconscious bias, the quote from Wieman is likely just as true in computer science.

Unconscious bias is a factor in women’s underrepresentation in STEM generally, and computer science specifically. The idea is that we all have biases that influence how we make decisions. Unconsciously, many of us (at least in the Western world) are biased to think computer scientists are mostly male. Unless we consciously recognize our biases, we are likely to express them in our decisions. A 2013 multi-institutional study (<http://bit.ly/2jUJ9p>) found undergraduates see computer scientists as male. That’s a source for bias.

Women in computer science (CS) report on biases that keep them from succeeding in computer science (<http://bit.ly/2BH6N9P>). Studies show female science students are more likely to be interrupted and less likely to get instructors to pay attention (<http://for.tn/2A7Zilu>). The National

Center for Women and IT (NCWIT) has developed a video titled “Unconscious bias and why it matters for women and tech” (<http://bit.ly/2zPxyHW>). A recent report from Google and researchers at Stanford University (<http://bit.ly/2A8WiPL>) presents evidence that unconscious bias influences teachers’ decisions in CS classrooms; they recommend professional development for the teachers, to help reduce their expression of bias. Google is funding the development of a simulation for teachers to address unconscious bias (<http://bit.ly/2jhpEkp>).

The tech industry recognizes unconscious bias is a significant problem. Microsoft is making its unconscious bias training available worldwide (<http://bit.ly/2AsUOyu>). Google is asking 60,000 employees to train to recognize unconscious bias (<http://read.bi/2kp144m>).

So here’s the question: If unconscious bias is pervasive in computing, and training is our best remedy, how can untrained students evaluate their CS teachers without bias?

Computing Research News raised concerns about bias in student evaluations of CS teaching in 2003 (<http://bit.ly/2koz7tk>). A recent study found students biased against female instructors (<http://bit.ly/2AVRdJZ>). There is evidence online students evaluate instructors more highly if they think they are male (<http://bit.ly/2AZuk95>).

I have not seen a study showing bias in CS students’ evaluations of their teachers, but the evidence is pretty overwhelming it’s there. How could the students avoid it? We know without training, students evaluate teachers with bias. We have found unconscious bias across computing. How could undergraduates evaluate a female CS instructor fairly? What might lead them to evaluate teaching without gender bias?

We have too few women in computer science. We need to recruit more female faculty in CS and retain them. We need to encourage and reward good teaching. Biased student evaluations as our *only* way to measure undergraduate teaching quality doesn’t help us with either need. ■

John Arquilla is professor and chair of defense analysis at the U.S. Naval Postgraduate School; the views expressed are his alone. **Mark Guzdial** is a professor in the College of Computing at Georgia Institute of Technology.

© 2018 ACM 0001-0782/18/2 \$15.00

Quantum Technology Forgoes Unconditional Security to Extend Its Reach

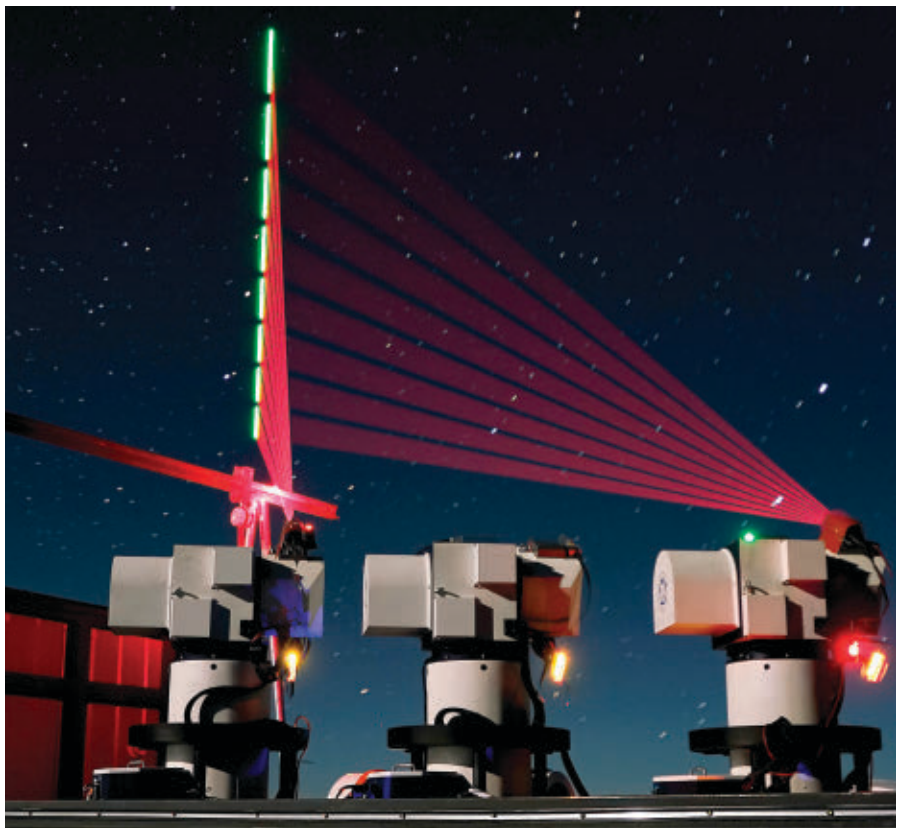
Two projects in China demonstrate the possibility of global quantum key distribution networks.

THE SUMMER OF 2017 saw two Chinese projects demonstrate the potential to move encryption based on quantum techniques from small-scale trials into networks that can span the globe.

The completion of an \$85-million fiber-optic network that runs 1,200 miles from Beijing to Shanghai through Jinan and Hefei marks a distance record for quantum key distribution (QKD). Designed to overcome a fundamental problem in conventional cryptography—how to transmit a private encryption key securely to another user without risk of it being intercepted—QKD has until recently been limited to networks that span no more than a few hundred miles.

As the Beijing-Shanghai link started to transmit QKD-protected data, another team based in China reported the successful use of a satellite to transfer similarly encrypted data transferred between ground stations 750 miles apart.

Grégoire Ribordy, CEO of Geneva, Switzerland-based ID Quantique, says of the satellite experiment, “Everybody



This composite photo taken on Dec. 10, 2016 shows a quantum communication ground station in southwest China's Tibet Autonomous Region.

IMAGE BY XINHUA/JIN LIWANG VIA GETTY IMAGES

in the science field knew it was possible, but there is a difference between knowing and seeing a demonstration. This is a technology that makes possible global QKD networks.”

The concept underpinning QKD is now more than half a century old, although it spent the first 20 years in obscurity. The original proposal was rejected by editors working on a journal published by the Institute of Electrical and Electronics Engineers (IEEE) in the 1970s. It was resurrected as the basis for a protocol devised by Charles Bennett and Gilles Brassard and presented at a 1984 IEEE conference on signal processing in New York.

The BB84 protocol uses the Heisenberg uncertainty principle, which argues measuring one part of a quantum state makes it impossible to determine other linked properties. Many BB84-based schemes use polarized light for this purpose. If a polarizer is aligned at angle of 45° , there is a 50% probability of the photon passing through being detected. Even so, its polarization is irrevocably realigned so further measurements cannot reveal its original state.

If the alignment is known, an eavesdropper can easily interpose their own detectors and copy the photon states successfully. But if the sender randomly switches alignments between perpendicular and angled positions, the eavesdropper has to guess correctly to be sure of passing the correct state on to the receiver. The QKD protocol has the sender and receiver check with each other which bits were transmitted successfully. If there are no physical weaknesses in the equipment used, the users can perform the check without disclosing bits in the final key. A low success rate for tested bits indicates an eavesdropper is present; at that point the parties can decide to try again or use a different channel.

Although the basic BB84 protocol relies on the ability to detect eavesdropping, variants of the protocol remove this requirement. In 2014, researchers at Toshiba designed a protocol that uses a second wave of selection on the sifted bits to yield a key that, under available technology, is guaranteed to be secure.

Numerous experiments into QKD have been conducted since the early 2000s to test the technology's reliabil-

2017 marked the 10th anniversary of the first use of quantum key distribution to protect voting data in elections in Switzerland.

ity and performance. Ribordy says 2017 marked the 10th anniversary of the first use of QKD to protect voting data in elections in the company's home country of Switzerland. He says, as found by the Chinese project, banks have been the primary early adopters of QKD, typically to support the backup of data between nearby datacenters.

The main problem with QKD is its range; photons are readily absorbed in long-distance fiber. The problem is less acute with atmospheric transmissions, as the satellite experiment demonstrated, but the losses increase exponentially in the time it takes to successfully send a single key bit, as they do with longer distances.

According to Pan Jian-Wei, a quantum physicist at the University of Science and Technology of China and leader of the satellite project, it would take hundreds of years to send a single key bit successfully from Beijing to Shanghai using a single fiber connection. The maximum practical range for a fiber link is less than 100 miles, and the eastern China network employs more than 30 “trusted” nodes, each of which decodes and stores keys locally before initiating a QKD session with its nearest neighbor.

Even with short-range transmissions, the rate at which keys can be transmitted remains orders of magnitudes below the rates needed to transfer bulk data. Ideally, QKD would be used to support the concept of the one-time pad, where no key bit is ever used twice. This reduces the probability of decryption using brute-force methods practically to zero. However, low practical key rates

mean that practical applications use the same key repeatedly to support the transmission speeds needed for commercial applications. Ribordy says a typical use-case is to use QKD to send frequent key updates to support symmetric ciphers such as AES. “You update the key, say, 10 times per minute. Very basically, you limit the vulnerability of a particular key being broken,” he explains.

John Leiseboer, CTO of Australia-based QuintessenceLabs, says: “When it comes to the one-time pad, I think that's a great place to go. But then again, I think, do we have to get to the one-time pad? It's better if there is no practical attack possible at all, but there are always practical things you need to take into account. You will always be open to weaknesses. I work with ways of trying to engineer a solution and work within the bounds of what's feasible and practical.”

The need to decode and store key bits at each trusted node increases the vulnerability of QKD-based networks, though these risks are no higher than they are for existing networks owned by telecom operators. Ribordy says, “If you think about telecom operators deploying QKD networks based on trusted nodes, they already have suitable secure facilities. It is relatively easy to guarantee the physical security assumption.”

One way to reduce the risk of relying on trusted nodes is to introduce quantum repeaters at each relay point. Such repeaters do not need to resolve the quantum states of photons as they are received. They allow the signal to be refreshed and passed on directly without having to store the key bits temporarily. Researchers have built devices that may be able to act as quantum repeaters, but they are still far from practical and would demand a change of equipment to use quantum entanglement to encode data. This is more difficult to achieve compared to the polarization-based technology of today's single-photon QKD systems.

A set of repeaters can, in principle, swap entangled states between pairs of photons that were created on separate links. By performing multiple swapping procedures, repeaters extend an entanglement across the entire

length of the network. The swapping is achieved at the cost of further drops in data rates, because not every attempt to swap is successful. However, the performance penalty is likely to be far lower than that encountered with longer runs of fiber.

"I would dearly love to see a practical quantum repeater developed," Leiseboer says. "It would be a significant game changer. But it's a relatively long-term option, and we don't know when or whether it's going to happen. Until we get there, we have to use other means."

Ribordy adds, "Unconditional security through quantum repeaters would be nicer to achieve. But for the first generation of deployment, it's feasible to use trusted nodes combined with satellites. We can deploy optical fiber on continents, and then bridge across continents using satellites. That enables the first wave of quantum communications."

By demonstrating QKD working over longer distances, the Chinese projects have helped reinvigorate interest in the technology, Ribordy says.

In February 2017, six months after the launch of the Chinese Micius satel-

lite used to perform quantum-communications projects, the European Space Agency (ESA) held a meeting to hear pitches for its ScyLight satellite-based QKD project. "It's a bit too bad that we had to wait for the Chinese to do it to pique the interest of the ESA in QKD," laments Ribordy.

Both ID Quantique and Quintessence Labs are working with the space and defense agencies of Switzerland and Australia, respectively, on their own experiments into space-borne QKD networks. Work on QKD has yielded other results that will be useful across cryptography, whether it is based on quantum or classical communications.

"We are developing techniques and subsystems that can be used now in real products that are not tied to QKD. For example, we now have a quantum random-number source that's entirely as a result of the QKD effort," Leiseboer says.

The next steps for QKD proponents are to build on the existing experiments to demonstrate the effectiveness of QKD and to start bringing the cost of the relay equipment down. Leiseboer notes, "There is no short-term

major breakthrough that needs to be achieved. It's about reducing the cost and making it smaller." 

Further Reading

Bennett, C. H. and Brassard, G. Quantum cryptography: Public key distribution and coin tossing *Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing*, Vol 175, p 8. New York (1984)

Liao, S.-K. et al. Satellite-to-ground quantum key distribution *Nature (Accelerated Preview)*, August 2017). DOI: 10.1038/nature23655

Sasaki, T., Yamamoto, Y., and Koashi, M. Practical quantum key distribution protocol without monitoring signal disturbance *Nature*, Vol 509, p475 (2014). DOI: 10.1038/nature13303

Pirandola, S., Laurenza, R., Ottaviani, C., and Banchi, L. Fundamental limits of repeaterless quantum communications *Nature Communications*, 8:15043 (2016). DOI: 10.1038/ncomms15043

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

© 2018 ACM 0001-0782/18/2 \$15.00

Milestones

Mustafa Prize Recognizes Work by ACM Fellow

One of the Islamic world's most prestigious science and technology awards, the Mustafa Prize, was presented recently to Erol Gelenbe, a professor in the Department of Electrical and Electronic Engineering at Imperial College London and an ACM Fellow.

In Iran's capital, Tehran, Gelenbe was presented with \$500,000, along with a special medal and certificate.

Said Gelenbe, "I am surprised and honored to get this award. I've never been to Iran before and this promises to be an amazing first visit. I've always had a passion for fundamental research on the mathematical foundations for computer and communication systems, which lead to building better systems and improving their usage and performance, and that is what has driven me over the years, so it is nice to be honored in this way."

Gelenbe has been a prolific researcher, publishing in excess of 360 papers through the course of his career in Belgium, France, the U.S., and the U.K.

He is the inventor of the eponymous G-networks or Gelenbe Networks that are the underpinning science used to evaluate the performance of computer networks, while they are controlled to ensure they function smoothly without overloading.

In one of his many breakthroughs, Gelenbe invented a random neural network that closely models the spiking behavior of natural neuronal systems, together with learning algorithms for these systems.

Gelenbe and his colleagues are also credited with inventing a patented multi-hop computer architecture for conveying voice and images over computer

networks. He also established the theoretical underpinnings for this approach, which is now widely used in the telecommunications industry.

For some of his inventions, Gelenbe and the teams he created gave up personal financial reward in exchange for distributing software freely to universities worldwide, so as to spread knowledge and social benefit.

During his long career, Gelenbe has supervised more than 75 Ph.D.'s who now work in Asia, Africa, Europe, and the Americas. The American Mathematical Society ranks him among the top 30 most prolific Ph.D. supervisors in the mathematical sciences. His former Ph.D. students include many talented women researchers and academic leaders.

Eric Yeatman, head of the Imperial College London

Department of Electrical and Electronic Engineering, said, "The underpinning technologies that enable our communications and ICT systems to be efficient and seamless are, in significant part, thanks to the work of Erol. We are very proud of his accomplishments, and on behalf of the whole department I send him my warm congratulations for this prestigious honor."

The Mustafa Prize is a science and technology award granted to top researchers and scientists of the Islamic world biennially. The Prize is awarded in four categories: Life Sciences and Medical Sciences, Nano Science and Nanotechnologies, Information and Communication Science and Technologies, and Top Scientific Achievement in other fields.

Going Serverless

Serverless computing lets businesses and application developers focus on the program they need to run, without worrying about the machine on which it runs, or the resources it requires.

PEOPLE LONG AGO got used to the concept of cloud computing; they would turn over their computational needs to a service provider—an Amazon or Microsoft—and no longer have to deal with the expense of buying and maintaining their own servers. However, they still had to determine what resources they needed, such as CPU time and memory.

Now, though, there is a newer approach that puts even more conceptual distance between the software and the machine that runs it. Serverless computing is meant to let businesses and application developers focus on the program they need to run and not worry at all about the machine it is on or the resources it requires. This higher level of abstraction is designed to make life easier for developers while it makes more efficient use of the cloud providers' infrastructure.

Serverless computing is more abstract than a virtual machine, which emulates a complete computer system inside another computer. "A virtual machine still has the word 'machine', so you still have the concept that you have RAM that is yours and you have drivers in a virtual machine that can get to the hardware," says Paul Brenner, associate director of high-performance computing at the University of Notre Dame, Indiana. Even with virtualization, setting up a virtual machine still requires a lot of planning. "You still had to think about how the network connects and how the switches connect and all that. You effectively still were building computer systems, just the pieces and parts came out of a cloud portfolio," Brenner says. "Serverless takes it a step further, where you don't even think about the infrastructure. You think of functions that your code needs to perform."

Reducing software to functions makes it easier for developers to write



apps. Say, for instance, the app needs to open an image. With serverless computing, the developer does not need to know where in the cloud the image is stored or how much memory it requires. "You want to put a funny hat on a kitten in your application, it just goes out to the cloud and does it," Brenner says. "You just hit 'function: add hat to kitten' and the function will go out and do it."

Serverless computing is just the latest option in cloud computing's "as a service" model, says Tim Hockin, principal software engineer at Google Cloud. Different versions of the model—infrastructure as a service, software as a service, and platform as a service—have cloud providers offering customers different levels of resources, from software subscriptions to full computing systems. Serverless is a fourth category: functions as a service. "Functions as a service is even more magical. You don't even think about the runtime that you're using. You just write a blob of code and we will run it for you," Hockin says. "You have to use our language, and you have to use it in a way that we are okay with; you have to use the libraries we offer. But if you use it in those bounds, man, your life is easy."

All the major cloud providers offer serverless computing. Amazon offers Lambda. Google has Cloud Functions. IBM has OpenWhisk. Microsoft has Azure. Serverless functions generally run for only microseconds, and must be written in a small number of languages, which include JavaScript and Python. The functions run only when triggered, for instance by an app asking for an image, and customers pay only for the time the code is running. As more triggers come in, the cloud provider runs more versions of the function, allowing it to scale up quickly without the developer having to figure out in advance how much memory or CPU time will be needed.

Moving Fast

Serverless is popular, Hockin says, because it reduces the time needed to get an application out to users. Developers want to be able to put out an app like Pokemon Go or Snapchat quickly to attract users, then improve its performance once it catches on. "Getting your thing out the door is way more important than optimizing it," Hockin says. "That comes with higher levels of abstraction. Higher levels of abstraction require you to be more divorced from the details."

The approach is well suited to workloads that have a burst of activity in a short time, says Ali Kanso, a senior cloud engineer at IBM's Thomas J. Watson Research Center in Yorktown Heights, NY. It could be useful for online ticket sales, for instance. The seller's website might have long periods of inactivity, but see thousands of transactions in the few minutes it takes for a Beyoncé concert to sell out. Without serverless, the seller might have to reserve and pay for resources that mostly sit idle, or watch the system crash when it is overwhelmed by a burst of traffic.

Serverless allows the seller to quickly grab additional computing resources in the small increments necessary for each ticket sale, then just as quickly shut them down.

It is also a good way to handle the demands of Internet of Things devices, Brenner says. Such devices are usually inexpensive, so they have simple hardware, which requires minimal software. Often their job is to take a sensor reading or capture an image and upload it to the cloud at intervals ranging from minutes to hours. Such short, sporadic activity fits the small, discrete functions of serverless.

Contain Yourself

Serverless computing is based on another, older concept: containerization. Containers are simplified versions of virtual machines, providing an environment inside which a piece of software can run. Brenner calls a container “a sandbox for software” that does not give the user access to the computer's hardware. Launching a virtual machine means loading in the operating system and all the libraries, and the process can take minutes; a container can be launched in less than a second, and the code copied into it in less than a second, so it's up and running quickly.

Containers allow services such as search and mail to run as quickly as they do, says Aparna Sinha, product management lead at Google Cloud. Google originally developed a container management system called Borg back in the early 2000s. Later, the company developed Kubernetes, an open source container system based on Linux. In 2013 another company, Docker, created its own open source container system for general use.

“Every large-scale operator in industry that's deploying web services, they're using some kind of containerization technology,” says Remzi Arpaci-Dusseau, a professor of computer science who studies distributed systems at the University of Wisconsin in Madison. “What's nice about Docker and the more-general open containers is they're more accessible to everybody, because they're free and open source.”

Every time a function is triggered, the system creates a container in which to run it. While running the function may take only microseconds, launching the container takes a second or two.

“You have to use our language, and you have to use it in a way that we are okay with; you have to use the libraries we offer. But if you use it in those bounds, life is easy.”

For many applications, that is fast enough, especially compared to creating a virtual machine. However, in some cases, Kanso says, even a second or two can be too long to wait. For instance, an app dealing with real-time stock market transactions, which can take place on the order of milliseconds, could not use containers. If an app deals with a series of events, each of which takes a couple of seconds, latency will keep increasing. Researchers developing container technology will have to figure out how to reduce the launch time, he says. “It appears to be one of the next critical questions.”

Arpaci-Dusseau developed OpenLambda, a research platform to tackle questions about serverless, along with several colleagues, including his former Ph.D. student Tyler Harter, now a software engineer at Microsoft. Getting containers to launch faster will be a challenge, they say. “If you really want to get down to say being able to start containers in 1 or 2 ms, we're going to have to make changes to Linux itself,” Harter says.

Systems using long-running programs take some time to initialize, but then improve by caching pieces of data near the processor. It is not clear how to use caching in serverless, where small operations run briefly and may be spread out on different servers, says Arpaci-Dusseau, but researchers are trying to figure it out.

Another challenge for serverless, Hockin says, is that many people want their apps to work with their databases. While containers work easily with stateless workloads, which do not re-

tain data, a database has to maintain its state over time, which conflicts with the here-and-gone nature of containers. Google has developed a method to capture the requirements of a stateful workload and turn them into application programming interfaces that allow users to manage their databases in a serverless setting.

Serverless is also not optimal for deep learning applications, or anything that requires large amounts of data or is designed to run for a long time, Brenner says. Hockin, though, believes containers and serverless computing can be useful for just about any type of application. “I think there will be coverage for every major class of application,” he says. “If there's people who want to do things, the technology will adapt.” If, for instance, some applications need lower latency than is currently available, researchers will figure out how to provide that.

“We may not be there fully yet, but I think that if there's people who want to do it, they will find a way to do it,” Hockin says. “We'll make anything work that they're interested in.” **C**

Further Reading

Hendrickson, S., Sturdevant, S., Harter, T., Venkataramani, V., Arpaci-Dusseau, A.C., and Arpaci-Dusseau, R.H.

Serverless Computation with OpenLambda, Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing, 2016.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J.

Borg, Omega, and Kubernetes, Communications of the ACM, 59, 2016.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., and Suter, P.

Serverless Computing: Current Trends and Open Problems, ArXiv, 2017.

McGrath, G. and Brenner, P.R.

Serverless Computing: Design, Implementation, and Performance, IEEE 37th International Conference on Distributed Computing Systems Workshops, 2017.

Why Serverless Computing?

<https://www.lynda.com/IT-Infrastructure-tutorials/Why-serverless-computing/599623/638412-4.html>

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

The War Over the Value of Personal Data

In a world increasingly dependent on turning personal data into profits, it is unclear how much that data is actually worth.

DATA HAS BEEN called the “new oil,” and one reason for this is that personal data greases the wheels of our connected world. It powers wildly lucrative social media platforms like Facebook and Snapchat. Data makes online advertising super-targeted (and super-profitable) for Internet giants like Google, and data about online habits is highly lucrative for any brand selling online (which is most of them).

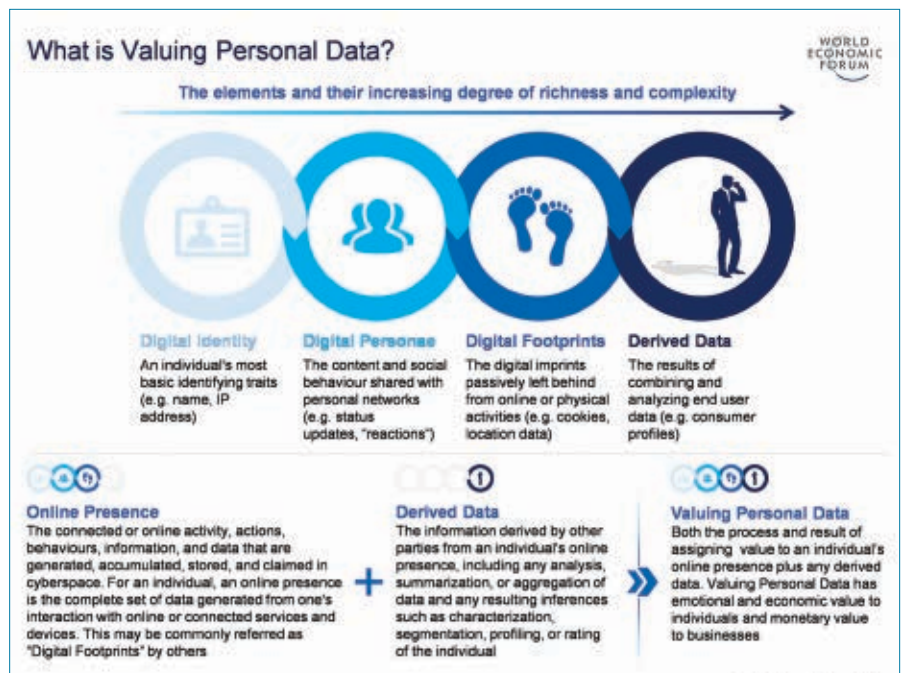
If data is the new oil, we’re discovering gushers each and every day. Consultancy IDC predicts the total amount of data generated globally will hit 44 zettabytes by 2020, a tenfold jump from 2013’s 4.4 zettabytes.

The value of this new oil has been enhanced by artificial intelligence (AI) and machine learning systems that are able to make sense of it all. As it turns out, machines are better at extracting value from structured and unstructured data than humans.

No matter how adept humans or machines are at giving data value, how much our data is actually worth remains an open question. In theory, data is everywhere, available to any company with the infrastructure to leverage it for commercial gain. The reality, however, is a little messier.

Individuals have uncertain control over how their data is collected, viewed, and monetized. Companies that want to earn profits from data, through AI or traditional data analysis, also face some obstacles.

“Companies, especially in industries such as financial services and health-care, have significant barriers to monetizing data, such as confidentiality, privacy, and regulatory requirements,” says Colton Jang, cofounder of LeapYear Technologies, a company that develops technology that enables firms to analyze and monetize sensitive data legally.



The result? A war is under way over data, but it's not entirely clear how much the resource is actually worth.

Quid Pro Quo

What is clear is that the volumes of data being generated are increasing. Cisco predicts a growth in annual Internet traffic of 175% over the next half-decade, from 1.2 zettabytes a year in 2016 to 3.3 zettabytes in 2021. While this data explosion is significant, it isn't clear how much of the data is “useful or valuable,” reports *Semiconductor Engineering*.

“A lot of industries have figured out that their business, product, and business models could be impacted by a different utilization of the data that is somehow attached to their devices or their business models,” Aart de Geus, chairman and co-CEO of software company Synopsys, told *Semiconductor Engineering*.

“If you can harness that in a way that finds shortcuts and efficiencies, or

just completely different ways of going about business, that is high-impact.”

This has firms in nearly every industry looking for ways to both acquire, and extract value, from consumer data. One of the major fronts in this is the home.

IDC predicts the number of devices connected to the Internet will triple by 2020 to 30 billion, and nearly triple again five years later. Many of these “Internet of Things” (IoT) devices are smart home appliances and tools, like connected refrigerators or general-purpose voice assistants like Amazon’s Echo device, which is powered by the company’s Alexa machine learning system.

IoT devices rely on data to drive user adoption. Your smart fridge may start with a series of assumptions about your buying habits based on past purchase data, then guess at what items to automatically reorder. To keep you happy, the makers of the fridge must continually ingest data on your habits

ACM Journal of Data and Information Quality



*Providing Research and Tools
for Better Data*

ACM JDIQ is a multi-disciplinary journal that attracts papers ranging from theoretical research to algorithmic solutions to empirical research to experiential evaluations. Its mission is to publish high impact articles contributing to the field of data and information quality (IQ).



For further information
or to submit your
manuscript,
visit jdiq.acm.org

to improve the quality of their guesses about what you want to buy next.

If they succeed, you keep using the fridge. If they do not, it's off to a competitor.

In all cases, data is the oil that powers the device. It is the resource that enables the device to improve—and thus attract more customers. This makes your personal data extremely valuable.

The question is: How valuable?

Consumers are potentially willing to give up their data—but they want something in return. For instance, a study conducted by Parks Associates found approximately half of U.S. households are willing to share smart device data and control in their home in exchange for a discount on electricity.

However, not every price is one consumers are willing to pay. The study also found that consumers were more likely to share their data in exchange for incentives like discounts, versus “intangibles” such as “product recommendations or simplified ordering.”

This seems like a pretty straightforward quid pro quo: companies need to make it worthwhile for consumers to relinquish their personal data. In the case of smart homes, maybe that's a discount on bills. For a social network like Facebook or an Internet giant like Google, users are given state-of-the-art communication and search tools in exchange for data on their browsing habits.

In some cases, companies and consumers agree on the value of data and transact in kind. But even when they do, turning consumer data into dollars can be difficult. The real value of data may actually lie in its aggregation. Tech titans are looking to learn about millions of people at once, not individuals, and they value data that has been analyzed in aggregate to deliver insights that can be monetized, rather than a muddle of machine-generated data that hasn't been assessed.

This leads to wildly different assessments of the monetary value data provides. Business analytics student Pauline Glikman and econometrics professor Nicolas Gladly tried to assess data's value in a 2015 *TechCrunch* article. In assessing Facebook acquisitions of WhatsApp and Instagram, as well as Microsoft's purchase of Minecraft, they found the value of each user was anywhere between \$15 and \$40. How-

ever, general information about individuals, like age or gender, was sold for as little as \$0.0007 per data point by data brokers that collected this information online.

So how much is data actually worth? The short answer is, it depends. This uncertainty introduces some complexities when governments attempt to introduce blanket data protection regulations.

Regulatory Hurdles

In the European Union, the General Data Protection Regulation (GDPR) will come into force in May. In the opinion of Jang at LeapYear Technologies, this law is the most significant one affecting individual and corporate data. “Any company that collects or processes data on EU citizens will need to comply with strict new requirements for data protection, or face massive fines for non-compliance,” he says.

The GDPR protects any information “that can be used to directly or indirectly identify the person,” according to the European Union's official website on the regulation. This includes “anything from a name, a photo, an email address, bank details, posts on social networking websites, medical information, or a computer IP address.”

The rights conferred by the GDPR to EU citizenry include the right to be notified of data breaches, visibility into how their personal data is used by companies that collect it, and the “right to be forgotten,” or the right to request that whom-ever has their personal data erase it.

This law gives individuals increased control over their personal data and, by extension, its value to firms who want to access it. Companies that sell to EU citizens will no longer be sure the data they collect will be available indefinitely. In fact, says Jang, companies that process large volumes of sensitive data, use data to predict or profile, and/or transfer sensitive data across borders should be preparing now.

“Companies will need to incorporate anonymization, data minimization, and privacy-by-design into their data processing activities,” Jang says. The GDPR can apply to companies outside the EU, which puts a large swath of firms at risk. In fact, any company that offers goods and services to EU citizens must comply with the new regulatory environment.

An Uneasy Peace

There may not need to be a war over the value of personal data; or, at least, not one that pits consumers directly against the firms to which they give their money.

Companies like LeapYear Technologies are helping to navigate the gap between the needs of companies and the regulations designed to protect individuals. LeapYear develops cryptographic technology that enables statistical analysis of a dataset without disclosing information about individual records. “Analysts can use our API to compute reports, statistics, and machine learning models against the data without being able to view or extract any information from the underlying data source,” Jang says.

This kind of compromise may be necessary. For one thing, it’s extremely difficult for companies in the connected era to comply fully with regulations like GDPR.

“Almost all data monetization strategies involve repurposing existing data assets that were originally collected for another purpose,” says Jang. “Under GDPR, this cannot be done without explicit opt-in consent from each citizen.”

On the other hand, it is easier than ever to collect data on a user’s online habits from anywhere in the world—in real time.

With cryptographically anonymized datasets like the ones LeapYear produces, consumers could enjoy the benefits of free online platforms and services without worrying about their data being abused. Brands could monetize that data and improve products with confidence, certain they won’t incur huge downside risk by doing so.

After all, the value of personal data is uncertain for companies precisely because the cost of noncompliance is oh-so-dear.

“Processing personal data in a non-compliant manner [under the GDPR] can result in significant fines, the greater of 20,000,000 Euros or 4% of global revenues,” says Jang. “Given the magnitude of the penalty, companies that collect or process data in the EU will need to reduce the scope of their data strategy or overhaul core business processes to incorporate new privacy-enhancing technology.”

On paper, consumer data is valuable

Companies that collect or process data in the EU will need to reduce the scope of their data strategy or incorporate privacy-enhancing technology into core business processes.

to firms that require it for better products and competitive advantages. In reality, much like the first rush of wildcaters and opportunists looking to find oil in unlikely places, Internet firms are finding the penalty for drilling in the wrong spot is costly indeed. **C**

Further Reading

Baar, A.
Smart Home Consumers Willing To Give In Order To Get
MediaPost, July 26, 2017
<http://bit.ly/2w2NSE9>

Glikman, P., and Gladys, N.
What’s the Value of Your Data?
TechCrunch, October 13, 2015
<http://tcrn.ch/2xBiBux>

Kanellos, M.
152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things,
Forbes, March 13, 2016
<http://bit.ly/2vnXGXg>

Li, C., Li, D.Y., Miklau, G., and Suciu, D.
A Theory of Pricing Private Data
Communications, December 2017,
<http://bit.ly/2j57NMQ>

Sperling, E.
The Rising Value of Data
Semiconductor Engineering, August 8, 2017
<http://bit.ly/2vf9qMz>

EU General Data Protection Regulation Portal
<http://www.eugdpr.org/>

Logan Kugler is a freelance technology writer based in Tampa, FL, USA. He has written for over 60 major publications.

© 2018 ACM 0001-0782/18/2 \$15.00

ACM Member News

TRACKING DOWN THE PHYSICAL INTERNET



“I have always been interested in math and puzzles,” says Paul Barford, professor of Computer

Science at the University of Wisconsin-Madison. “The path that led me to computer science was based on recognizing the mathematical foundation of computer science.”

Barford earned his B.S. in electrical engineering from the University of Illinois at Urbana-Champaign, then worked in industry for eight years before completing his Ph.D., in computer science at Boston University.

His research interests focus on collecting data and then applying analytic techniques to it. “I enjoy this very practical aspect of research,” Barford says. “I like to get my hands on equipment, and data.”

For the past six years, Barford has been working on the Internet Atlas Project, a comprehensive repository of the physical Internet being developed by the Center for Applied Internet Data Analysis. “I am trying to figure out where the Internet is actually deployed—the physical conduits—the places where those conduits are actually connected to communication devices,” he explains.

A colleague described his Internet Atlas work as “Internet archeology.” Barford says it was a lot of work to develop techniques to identify the physical paths of Internet data, as well as the nodes at which they terminate. His work on Internet topologies has allowed him to apply this data to understand how the Internet can be made more robust and secure.

“I am interested in how users are actually interacting with the Internet,” Paul adds. “At the end of the day, we are creating technology to serve users.”

—John Delaney

Inside Risks

Risks of Trusting the Physics of Sensors

Protecting the Internet of Things with embedded security.

SENSORS ARE TRANSDUCERS that translate the physical into the electrical. Computer software then interprets and operates on the binary representations rather than the direct physical or electrical quantities. For instance, drone software uses the abstraction of a signed integer to represent the output of a gyroscope for flight stability and attitude control.¹³ A *transduction attack* exploits a vulnerability in the physics of a sensor to manipulate its output or induce intentional errors. For example, malicious acoustic interference can influence the output of sensors trusted by software in systems ranging from smartphones to medical devices to autonomous vehicles. Autonomous systems should remain trustworthy despite untrustworthy components. Techniques from embedded security can help protect against analog threats to autonomous systems in the Internet of Things.

Threats. Thieves can break into cars using man-in-the-middle (MITM) attacks against keyless entry systems.⁵ Automotive manufacturers can neutralize MITM attacks with proper use

of cryptography. However, these MITM attacks exploit automotive systems that intend for radio waves to allow access. In contrast, transduction attacks use unintended functions of circuitry to threaten the integrity and availability of sensor output. Cryptography will not suffice to defend against transduction attacks. Attackers can exploit the physics of materials to fool sensors into becoming unintentional receivers of unwanted, malicious signals. The threat has grown such that the U.S. government warns manufacturers of transduction attacks that exploit the physics of sensors.¹

Sensors face two types of analog threats: opportunistic attacks requiring no special-purpose equipment, and advanced attacks that require special-purpose transmitters and basic understanding of physics. For instance, an opportunistic attack could use phishing to trick a person into playing untrustworthy music videos on a smartphone. The sound waves can influence the output of an accelerometer.¹⁴ Because a smartphone includes both a speaker and accelerometer, the adversary needs no transmitter or spe-

cial equipment to carry out this attack. An advanced attacker may build custom acoustic or radio frequency emitters. For instance, an adversary could use a Long-Range Acoustic Device (LRAD) to deliver intense sound waves from a mile away.

Vulnerabilities. Billions of deployed sensors lack designed-in protections against intentional physical manipulation.^{4,12–15} Most likely, the sensors were designed before the community understood the security risks. Researchers have repeatedly shown how an adversary can not only cause denial of service, but also control the sensor output itself with malicious analog signals at the resonant frequency of the sensor. Vulnerabilities tend to lurk deep within the physics of analog sensors. The risks bubble up into the software layer.

The DolphinAttack¹⁵ represents a transduction attack vulnerability whereby inaudible sounds can trick speech recognition systems into executing phantom commands. Microphones, especially miniature microelectromechanical systems (MEMS) microphones, can hear ultrasound. Although

the circuits and software attempt to attenuate such high-frequency sounds, an adversary can inject fake voice commands with ultrasound. The ultrasonic method exploits non-linear behavior within the signal path conditioning of the circuitry. The microphone is tricked into functioning as an unintentional acoustic demodulator. The DolphinAttack can silently manipulate almost all popular speech recognition systems, such as Siri, Google Now, Samsung S Voice, Huawei HiVoice, Cortana, Alexa, and the voice-controlled navigation system in an Audi automobile.

Malicious Back-Door Coupling. In the context of aircraft safety, front-door interference refers to unwanted signals that enter a system directly via an antenna port whereas back-door interference refers to unwanted signals that enter a system indirectly by coupling to its wires and other components.⁹ A transduction attack can use malicious back-door coupling to cause sensors to function as unintentional receivers and demodulators. That is, a sensor designed to sense one phenomenon (for example, deceleration of a car) may also accept unwanted signals (for example, sound waves at the resonant frequency of the sensor) without distinguishing the sources. Malicious back-door coupling can exploit a resonant frequency of unremarkable amplitude to overshadow a legitimate signal. There are many examples of malicious back-door coupling to violate sensor integrity. Malicious back-door radio waves tricked pacemakers into disabling pacing shocks.⁴ Malicious interference blending both front-door and back-door coupling fooled Tesla's sensors into hiding and spoofing obstacles,⁷ as shown in the three-image series in this column depicting real, spoofed, and jammed distances.

A hacker does not necessarily require special-purpose equipment to exploit back-door coupling in sensors. One could co-opt nearby software-controlled emitters common in laptop computers, smartphones, speaker systems, and even light bulbs. For instance, our research demonstrated how playing sounds embedded in a YouTube video allows an adversary to control the output of a smartphone's MEMS accelerometer. The exploit works because of mechanical coupling

Advanced sensor attacks. Sensors translate the physical into the electrical for interpretation by a computer system. However, analog signals can spoof data by exploiting the physics of sensors. This photo shows how malicious electromagnetic waves can trick software processing signals from a thermocouple into displaying an impossibly low temperature (-1409° F is 527° K below absolute zero).



between the sensor and the smartphone's built-in speaker that emits malicious signals modulated over a carrier at the resonant frequency of the sensor to induce a chosen sensor output.¹⁴

Trustworthy Embedded Systems

Protecting against transduction attacks is difficult because the consequences arise as software symptoms, but the risks begin in the physics of hardware. Good security practices such as static analysis, fuzz testing, and signed software updates are insufficient to protect against a sensor delivering intentionally false data. Software security tools were not designed to control for analog security risks. Thus, we recommend a return to classic engineering approaches for more trustworthy embedded systems to cope

with threats to the underlying physics of sensor technology.

- Shift from component-centric security to system-centric tolerance of untrustworthy components.

- Make the output of sensor hardware more continuously checkable by software for adversarial influence.

- Make attacks more difficult by manufacturing circuits in a manner to reduce effects of resonance.

Avoid component-centric security. Sensor systems should remain safe despite adversarial influence on untrustworthy components. Fault-tolerant systems pioneered the non-adversarial variant of this problem by limiting damage with techniques such as compartmentalization. However, faults and defects that develop after verification cannot be detected by verification. In computer security, the adversary controls the probability distribution of maliciously induced errors in components and can induce faults after verification.

Systems that treat security as just another component rather than a property will survive poorly against analog adversaries who can manipulate sensors with transduction attacks. Trusted components do not suffice to ensure a trustworthy system. For instance, a secure processor will happily sign false sensor data if blindly

Autonomous systems should remain trustworthy despite untrustworthy components.

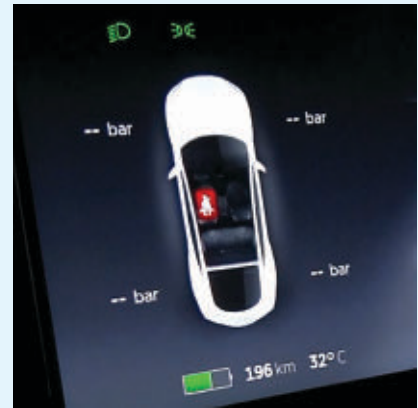
Malicious interference fooled Tesla's sensors into hiding and spoofing obstacles:⁷ (a) Real distance; (b) spoofed distance; (c) jammed distance.



(a)



(b)



(c)

accepting output from a trusted sensor rather than continuously doubting and checking trustworthiness of sensor output. Trustworthy components can fail catastrophically when attacks succeed; trustworthy systems can fail more gracefully when attacks succeed. Key to overall system trustworthiness is the ability for systems to check the trustworthiness of sensor output.

Make the security of sensor output continuously checkable. A central principle of information security⁸ is to consider inputs as circumspect until shown trustworthy (for example, by satisfying an independent check). Sensors may contain self-calibration circuits tested with injected signals during manufacture or power-up to verify the sensors perform as specified. Self-checking is difficult even when mother nature is the adversary. NOAA discovered its algorithms erroneously excluded output from a temperature sensor in Alaska because of a false positive of an anomaly detection algorithm.³ Sensors threatened by intentional transduction attacks must clear an even higher bar of continuous checkability.

Sensor interfaces could continuously convey additional evidence for applications to perform end-to-end checks of sensor trustworthiness. Some sensors already maintain debugging information internally, but do not expose the information across the hardware-software API. Sensors could expose spectral analytics, confidence indicators, or other hints such that software applications could better de-

tect threats such as signals at known resonant frequencies. A system can also compare data from multiple sensors operating on different physical principles (for example, comparing a reed switch and hall-effect sensor that sense magnetic fields). An engineering challenge is reconciling security with constraints of performance, board space, and cost. Exposing checkable hints of sensor output trustworthiness would enable a shift away from component-centric security toward system-centric security.

Specify physical security. When we reported an acoustic security flaw that allowed adversarial influence of accelerometer outputs, one manufacturer made an innovative recommendation that specifies how to more securely attach a sensor to a circuit board.² The response to the CERT report may be the first example of advising customers to physically manipulate a drill

bit, rather than digital bits, to mitigate a security vulnerability. Customers were advised to use inner mounting posts to a hard case to reduce board deflection near a sensor and ensure the vibrations of the board are above the resonant frequency of the sensor. Drilling holes differently in a circuit board can shift the resonant frequency out of the range that nearby acoustic transducers can generate or that the sensor's non-linearities can demodulate. The manufacturer also advised customers to place physical trenches around boards containing speakers to reduce mechanical coupling. Such simple, physical approaches can serve as effective compensating controls to decrease the risk of transduction attacks.

Embedded Security Education

Security is a system property. Thus, design of a sensor-driven, safety-critical system deserves supervision by a systems engineer with broad knowledge of computer security risks. Team leaders for such systems will need to master skills from physics, electrical engineering, and mechanical engineering to computer science, information science, public policy, and ethics.

Interdisciplinary teams. For medical devices and vehicles, an engineering team will minimally need a blend of experts from mechanical engineering, electrical engineering, and computer science who share an awareness of risks and recognize the value of working together. Students destined for solving these types of problems need

Cyberphysical systems must cope with analog threats that an adversary could exploit without any special-purpose equipment.

early exposure to interdisciplinary teamwork in classes and internships. However, not all engineers must master the underlying physics of computer security. Instead, every team member needs a basic awareness of the risks. A system always includes risks that will fall outside an individual team member's area of expertise. Thus, each engineer has an ethical responsibility to maintain awareness of analog security risks, inform management of uncontrolled risks, and know when to ask for expert help from a team leader.

The notion of interdisciplinary education is not new to computer science. In the 1990s, the software engineering community debated a shift toward interdisciplinary education beyond the confines of computer science.^{10,11} Similarly, a good engineer for embedded security will not simply be a good computer scientist or a good programmer. Interdisciplinary education and teamwork is key to ensuring security of sensor-driven, safety-critical systems.

Educational opportunities for embedded security. Aspiring system-security engineers need opportunities to learn fundamentals of embedded security. However, computer science curricula have little room to add material given the pressure to meet the industry's demand for gifted programmers. How can computer science programs create expert embedded security graduates under these constraints? Computer science cannot succeed alone.

Engineering schools should offer interdisciplinary educational programs for ambitious students to learn how to protect cyberphysical systems. Students would learn not just fundamentals of computer science and computer security, but also the physics of computational abstractions. A software engineer may take computer security courses to learn threat modeling, cryptography, and secure programming methodologies. To master the concepts and skills for embedded security, an engineer would also take courses that teach the fundamentals of signals and systems, communication theory, and classical physics. For instance, defending against transduction attacks involves spectral analysis, mechanical resonance, and modulation. Students wishing to become experts in embed-

ded security must understand how each layer of computation from sensors to human behavior can fail when subjected to adversarial interference.

Back to basics. Students are losing an appreciation for the physical machines that implement computational abstractions. Students graduating from departments that diminish the role of computing machinery will not be prepared to create trustworthy cyberphysical systems. For instance, students unaware of transduction attacks may falsely believe that verified software is failure-proof. Math-centric departments tend to avoid courses that emphasize building physical systems. If a department eliminates computer architecture, students may seek comfort hiding behind a beautiful Java facade rather than facing the ugly limitations of computing machinery. Even engineering-centric computer science departments succumb to this problem. Students may desire immediately marketable programming skills over understanding the fundamental limitations of the machines on which their software runs.

Students creating the next generation of trustworthy cyberphysical systems need an exposure to the physical limitations of the machines implementing each abstraction. An effective way to do this is to include labs featuring experiments of the kinds suggested earlier in this column. Tomorrow's software engineer must master both math-centric and engineering-centric skills while understanding the physical limitations of computational machinery. This topic deserves a longer conversation.

Conclusion

Sensors are vulnerable to spoofing by transduction attacks. Cyberphysical systems must cope with analog threats that an adversary could exploit without any special-purpose equipment. Automobiles decide whether to deploy an airbag based on data from accelerometers.¹⁴ Pacemakers and defibrillators decide whether to emit shocks based on data from cardiac sensors.⁶ It is inevitable and predictable that hackers will try to manipulate sensors to cause havoc. Autonomous systems making safety-critical decisions should remain safe even when an adversary can exploit

physics to influence the output of sensors. The community can reduce these risks by designing sensors to be continuously checkable for security properties and by increasing opportunities for students to master the physics of computer security and principles of embedded security. 

References

- Alert (ICS-ALERT-17-073-01A). MEMS Accelerometer Hardware Design Flaws (Update A), (Apr. 11, 2017); <http://bit.ly/2CjTdcD>.
- Analog Devices Advisory to ICS-ALERT-17-073-01 (Apr. 2017); <http://bit.ly/2EPF9cc>.
- Arndt, D. Alaskan North Slope climate change just outran one of our tools to measure it. (Dec. 6, 2017); <http://bit.ly/2AFNBjz>.
- Foo Kune, D. et al. Ghost Talk: Mitigating EMI signal injection attacks against analog sensors. In *Proceedings of IEEE Symposium on Security and Privacy* (Oakland, CA), May 2013.
- Francillon, A., Danev, B., and Capkun, S. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, The Internet Society, 2011.
- Fu, K. Pacemaker recall exposes national need for research and education in embedded security. In *Computing Community Consortium (CCC)*, (Sept. 2017); <http://bit.ly/2xBEgcl>.
- Liu, J., Yan, C., and Xu, W. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicles. In *DEFCON24* (Aug. 2016); <http://bit.ly/2EQNOLs>.
- Neumann, P.G. Fundamental trustworthiness principles. In *New Solutions for Cybersecurity*. In *MIT Press/Connection Science*, H. Shrobe, D. Shrier, A. Pentland, Eds., Cambridge, MA, 2018.
- Nguyen, T. Cumulative interference to aircraft radios from multiple portable electronic devices. In *IEEE Conference on Digital Avionics Systems*, 2005.
- Parnas, D.L. Education for computing professionals. In *IEEE Computer* 23, 1 (Jan. 1990), 17–22.
- Parnas, D.L. Software engineering programmes are not computer science programmes. In *Annals of Software Engineering* 6 (1998), 19–37. (Reprinted in *IEEE Software* (Nov./Dec. 1999), 19–30.
- Rouf, I. et al. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *Proceedings of USENIX Security Symposium*, (Aug. 2010).
- Son, S. et al. Rocking drones with intentional sound noise on gyroscopic sensors. In *Proceedings of USENIX Security Symposium* (Aug. 2015).
- Trippel, T. et al. WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. In *Proceedings of IEEE European Symposium on Security and Privacy (Euro S&P)*, (Apr. 2017); <http://bit.ly/2Cl2KQn>.
- Zhang, G. et al. DolphinAttack: Inaudible voice commands. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, October 2017.

Kevin Fu (kevinfu@umich.edu) is Associate Professor of Electrical Engineering and Computer Science at the University of Michigan.

Wenyuan Xu (wyxu@zju.edu.cn) is Professor and Chair of the Department of Systems Science and Engineering at Zhejiang University.

The authors thank Steve Bellovin, Robert Dick, Peter Denning, Nancy Leveson, Peter Neumann, David Parnas, Jerry Saltzer, Zeynep Tufekci, and Ben Zorn for their review comments.

This work is supported by NSF CNS-1330142. The views and conclusions contained in this column are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF.

Copyright held by authors.

Education

The Inclusive and Accessible Workplace

Maximizing the performance of neurodiverse talent.

AS COMPUTER SCIENCE (CS) learning opportunities expand across the U.S., related diversification efforts to make “CS for All” must include brain diversity. Neurologically different individuals, like those with *attention and learning disorders*, are often lost in conversations about broadening participation in computing. Yet their diverse experiences and perspectives are an asset to fields that require innovative thinking, like computing. Most CS learning and work settings have a long way to go to make the field more inclusive for people with these differences, and work practices accessible to them. If we are successful in growing CS educational opportunities to include diverse learners, the workplace must be ready to welcome and retain a neurodiverse talent pool. Employers need concrete management strategies that will maximize the performance of these current and future employees. Education research exploring ways to adjust teaching practices for CS students with varied attention and learning disorders can serve as a starting point. We refer to this diverse group of individuals as *neurodiverse talent*. Good teachers use inclusive classroom practices, adjust strategies as needed, and provide additional supports to help all students succeed. Managers looking to benefit from neurodiversity must similarly employ inclusive practices and adjust techniques to provide individual supports where needed.



Neurodiversity Is Often Misunderstood

The term *neurodiversity* is used to describe a spectrum of neurological differences, which result from a normal, expected range of variation in the human genome. Individuals with an “atypical” neurological configuration, such as with ADHD or autism spectrum, are referred to as *neurodivergent*. People with a “typical” neurological configuration (that is, con-

forming to what some would interpret as “normal”) are considered *neurotypical*. The term *neurodiverse* is used to describe a group of people with varied neurocognitive functioning. These terms are often confused.

Atypical individuals with attention and learning disorders experience “neurologically-based difficulties in reading, writing, math, organization, focus, listening comprehension, social skills, motor skills or a combination of

these.”³ They often demonstrate average to superior intelligence and possess varying strengths, yet experience unexpected underachievement in basic academic skills. While their ability to receive, store, process, retrieve, or communicate information may be affected, they can be successful when provided with accessible instruction and support. Neurodivergent individuals face a range of bias and misperceptions. In 2012, the National Center for Learning Disabilities conducted a survey, randomly sampling U.S. adults and found that 43% of respondents mistakenly think that learning disorders “are correlated with IQ.”¹

Researchers estimate that one in five children in the U.S. have attention and learning issues.³ At the postsecondary level, the largest group of students who report a disability are those with these less-visible disorders. In the workplace, 46% of working-age people with learning disorders (defined as 18–64 years) are employed, yet only 19% of them report disclosing their disorder to an employer.¹ This similarly occurs in postsecondary education, suggesting the current computing workplace is likely far more neurodiverse than we realize, and includes employees who would benefit from more inclusive and accessible operations and management techniques.

Neurodiversity Benefits Computing

The computing field has a lot to gain from the one in five youth in the U.S. with attention and learning disorders. For example, some suggest the best and brightest programmers are dyslexic (a reading disorder) because of their strengths in visualization, analytical and strategies reasoning, and hands-on problem solving.² As one CS student told us, “a lot of us have always been on the outside ... we can think on the outside ... that type of thing puts me and other kids with learning differences at an advantage ... we think in a more creative way.” But for those gains to be realized, the workplace must be inclusive of, and accessible to them.

The Americans with Disabilities Act (ADA), a federal civil rights law provides legal protections and requires employers to provide “reasonable accommodations” for employees with disabilities. Accommodating neurodi-

The computing field has a lot to gain from the one-in-five youth in the U.S. with attention and learning disorders.

verse talent (and others with a range of disabilities) is not just about “doing the right thing”—people with attention and learning disorders are legally protected under ADA. In this period of increased CS learning opportunities for all students, employers carry a responsibility to know about appropriate accommodations and small shifts in current management practices to support the diverse range of new talent entering the workforce. Luckily, many resources that provide workplace guidance already exist.^a We can also look to large tech companies who have already started to adjust hiring and operational practices to accommodate neurodiverse talent (particularly those on the autism spectrum), like Microsoft and SAP. Employers will find the monetary investment of accommodations is generally small, if anything. Approximately half of accommodations have no cost,^b like shifts in management techniques to leverage talent.

Education Research to Inform Workplace Adjustments

Attention and learning disorders are life-long conditions (that is, students do not outgrow them). Thus, strategies to include and support these individuals are needed in both educational and workplace settings. Insights from education research in this area can inform approaches to maximize contributions from neurodiverse talent.

In our National Science Foundation supported research exploring strategies to make high school AP Computer Science Principles (CSP) more accessible to students with at-

tention and learning disorders (a partnership between Outlier Research & Evaluation at the University of Chicago, and Wolcott School), we identify general instructional practice adjustments, and lesson-specific adjustments that improve accessibility of, and student engagement in CS learning.^c Over the 2016–2017 school year, we regularly interviewed 12 students to understand how their attention and learning disorders related to their experience with various aspects of the class. Their responses directly informed our recommendations. Some of the challenges we heard students express included those related to **presentation** (“It was hard to read through the instructions handout [so] I had some of my classmates read [them] to me”), **response** (“If I have something in my brain, it’s hard for me to put it in the right order and get it out of my brain [to] communicate”), **timing** (“It was kind of rushed ... I would have been able to do better and think it through more if I had more time, especially ... the first time”), **physical settings** (“It’s a pretty large group ... It’s pretty distracting ... If I put in my headphones ... I can focus on my work”), and **social interactions** (“I got extremely frustrated cause they didn’t know what they were doing ... they thought they did everything right even though I blatantly told them ‘you messed up’”). All of our adjustment recommendations address at least one of five general categories:

- **Presentation:** Providing access to materials in multiple ways.

- **Response:** Providing options for solving or organizing work in alternate ways, and for demonstrating understanding.

- **Timing:** Offering additional time for tasks, projects, and assessments.

- **Settings:** Offering physical setting adjustments.

- **Social interactions:** Providing supports to help maintain focus and engagement in collaborative work and share concrete examples of collaboration.

These are the key areas where managers may also need to adjust current practices. Research suggests that teaching adjustments made to accommodate those with attention and

a <http://bit.ly/2D0AYcY>

b <http://bit.ly/2BuknO7>

c <http://bit.ly/2BGTKIp>

learning disorders often result in better academic success for *all individuals*.

Strategies for Increasing Contributions from Neurodiverse Talent

While there are many resources that provide specific strategy suggestions to improve accessibility of work environments, we offer a synthesis of these recommendations^{d,4} and our own from experiences supporting neurologically different youth and adults. While particularly well-suited for neurodivergent employees, these strategies are also beneficial for the range of neurotypical employees in a computing-related setting.

We realize there is considerable challenge for managers in recognizing which employees might need accommodations when so few disclose their differences. One great approach in this situation is to incorporate suggested strategies into all regular team operations, to create a workplace more inclusive of a range of workers. Shifting management approaches requires effort and flexibility, but the benefits to both employees and employers are considerable. Inclusive practices create an environment that allows employers to tap into, and acknowledge a range of perspectives and experiences, which are at the heart of innovation. As managers shift, neurodiverse talent must also take responsibility for establishing and sustaining their own practices to ensure success, to self-advocate when supports are needed, and to implement additional strategies to increase their focus and productivity.

Here, we share management strategy recommendations to maximize the strengths and performance of a neurodiverse workforce:

Present instructions and expectations both verbally and in writing to avoid ambiguity, support employees with memory deficits, and explicitly outline expectations.

- Provide project details and work tasks in both verbal and written communications.

- Clearly describe key expectations and instructions in writing for employees to revisit as needed.

Break down tasks and identify spe-

Employers, like classroom teachers, are responsible for adjusting practices to meet the needs of neurodiverse talent and providing workers with appropriate tools for success.

cific goals to support organization, prioritization, and time management of work assignments.

- Provide task checklists; Promote use of collaborative project management and time management systems to separate tasks for completion, and to keep track of time on tasks.

- Share flowcharts to describe steps of complicated processes and appropriate completion time.

Schedule frequent check-in meetings between employee and supervisor to provide time for direct communication and specific feedback.

- Establish a structure for weekly check-in meetings to keep work on track and clarify any misunderstandings.

- Provide personalized training and job mentors to support targeted areas of improvement.

Recognize the hurdles email writing presents for some employees.

- Grant sufficient time for writing and editing email communications.

- Offer editing options, like the use of text-to-speech software (to listen to their own writing), or support from co-workers to proofread email content and subject lines.

Permit employees to “pass” on note taking and on-the-spot idea generation to minimize anxiety from spelling and writing, and social or communication-related challenges.

- Be flexible with group note-taking duties in meetings (whiteboards;

shared electronic documents).

- Provide advance notice of meeting topics, and allow employees to submit ideas and feedback in writing, pre-, or post-meeting.

Offer flexible workspace arrangements to support organizational and concentration in open work environments.

- Enable employees to organize their personal workspaces and approaches to best fit their strengths and their own organizational needs.

- Designate quiet/private spaces that employees can use throughout the day to minimize distractions and focus.

Organizational commitment to utilizing management techniques like these requires dedication, person power, and time, yet these simple strategies are a low, or no-cost starting point to support workers with attention and learning disorders. In fact, some of these approaches are likely already used by effective managers. Employers, like classroom teachers, are responsible for adjusting practices to meet the needs of neurodiverse talent and providing workers with appropriate tools for success. Using strategies to support these workers (and make work less stressful for them) will ultimately maximize the contributions of *all* employees, improve team efficiency and productivity, and increase retention of great workers. **□**

References

1. Cortiella, C. and Horowitz, S. *The State of Learning Disabilities: Facts, Trends and Emerging Issues*. National Center for Learning Disabilities, NY, 2014; <http://bit.ly/2kfoDtA>
2. Dyslexic Advantage Team. *Dyslexia and computer programmers*. (Sept. 2016); <http://bit.ly/2kGYiEh>
3. Horowitz, S, Rawe, J., and Whittaker, M. *The State of Learning Disabilities: Understanding the 1 in 5*. National Center for Learning Disabilities, NY, 2017; <http://bit.ly/2oIJuPe>
4. Morris, M.R., Begel, A., and Wiedermann, B. Understanding the challenges faced by neurodiverse software engineering employees: Towards a more inclusive and productive technical workforce. In *ASSETS '15* (Oct. 26–28, 2015, Lisbon, Portugal), 173–184; DOI: <http://dx.doi.org/10.1145/2700648.2809841>

Sarah Wille (swille@uchicago.edu) is a Senior Research Scientist and Director of Computer Science Education Research for Outlier Research & Evaluation at UChicago STEM Education, University of Chicago, and a member of UChicago STEM Ed's Diversity, Equity, and Inclusion Committee.

Daphne Sajous-Brady (dsajous-brady@wolcottschool.org) is the Director of Student Services and Lead Teacher of the Learning Strategies Department at Wolcott School.

This material includes work supported by the National Science Foundation under grant number CNS-1542963.

Copyright held by authors.

^d <http://bit.ly/2BG1MRQ>



Kode Vicious

Reducing the Attack Surface

Sometimes you can give the monkey a less-dangerous club.

Dear KV,

My group is working on a piece of software that has several debugging features. The code will be part of a motor-control system when we are done. One of the features we have added is a small CLI (command-line interpreter) that we can use to change the parameters that control the motor and to see what effect these changes have on power consumption, heat, and other values tracked by our software. We first added the CLI just for our small group, but now we have found that both the QA and factory teams have come to depend on having it in place to do testing (QA) and preshipping checks in the factory.

As you might imagine, the ability to change the parameters of the motor once it has been shipped could lead to problems such as overheating, as well as a catastrophic failure of the motor. Even though our product is not meant to be some sort of IoT (Internet of Things) device, we do have a network connection available on our higher-end products so that the performance and wear on our motors can be measured over a network in the field.

I have told the QA and factory teams that there is no way we should leave this code in our shipping product because of the risks that the code would pose if an attacker could access it. They say the code is now too important to the product and have asked us to secure access to it in some way. Networked access to



the device is provided only over a TLS (Transport Layer Security) link, and management now thinks we ought to provide a secure shell link to the CLI as well. Personally, I would rather just rip out all this code and pretend it never existed. Is there a middle path that will make the system secure but allow the QA and factory teams to have what they are now demanding?

CLI of Convenience

Dear CLI,

See earlier editions of KV to find my comments on prototypes, because they are relevant here (for example, “Beautiful Code Exists, If You Know Where to Look”; <http://bit.ly/2C64HR2>). The problem is that once you give a monkey a club, he is going to hit you with it if you try to take it away from him. The CLI you and your team have created is a nasty-looking

club, and I would hate to get whacked with it.

The best way to reduce the attack surface of a piece of software is to remove any unnecessary code. Since you now have two teams demanding that you leave in the code, it is probably time to think about making two different versions of your binary. The application sounds like it is an embedded system, so I will guess it is written in C and take it from there.

The traditional way to include or exclude code features in C is via the prolific use of the `#define/#ifdef/#endif` preprocessor macros and abuse of makefiles. The first thing to do is to split the CLI functions into two sets: readers and writers. The readers are all the functions that return values from the system, such as motor speed and temperature. The writers are all the functions that allow someone to modify the system's parameters. The CLI itself, including all the command-line editing and history functions, is its own piece of code. Each module is kept under an `#if/#endif` pair such as this:

```
if defined(CLI_WRITER)
/* XXX Dangerous Code,
do not ship! */
endif
```

CLI_WRITER should be defined only via the build system and never as a define in the code. You are liable to forget that you defined the value during some of your own testing or debugging, and commit your fixed code with the value defined.

With the code thus segmented, you now define two versions of your binary: TEST and SHIP. The TEST version has all the code, including the readers, the writers, and the CLI itself. The TEST version can also have any and all debug functions that the QA and factory teams want to have prior to shipping.

The SHIP version of the code has none of the debug features and only the reader module for the CLI. I would say it goes without saying that the CLI must not have a `system()`-like function that allows the execution of arbitrary code. I would love to believe that could go without saying, but, guess what, I said it because I have seen too

The best way to reduce the attack surface of a piece of software is to remove any unnecessary code.

many systems with a “secure” CLI that contains a `system()` function.

If at all possible, you should link all of your binaries statically, without using dynamic libraries or KLDs (kernel-loadable modules). Allowing for dynamically loadable code has two downsides. The first downside is that some monkey can come along later and re-add your writer functions to the system. The second downside is that you lose your protection against someone accidentally leaving in a call to a writer function when they should not. In a statically linked binary, all symbol references must be resolved during the linking phase. If someone leaves a stray call to a writer function somewhere in the code, this error will be flagged at link time, a final binary will not be produced, and you will not be able to ship a polluted binary accidentally.

In each of the reader, writer, and CLI modules you should place a specially named symbol that will remain in the final binary. Pick obvious names such as `cli_reader_mod`, `cli_writer_mod`, and `cli_mod`. Before any binary is shipped, either placed into a device at the factory or put up on the company's software-update server, a release script must be run to ensure the `cli_writer_mod` symbol is not present in the shipping binary. The release script could look for a known function in the writer module, but programmers often like to change the names of functions, so adding a special symbol is easier and it is unlikely to change. For double extra bonus points, you can also have a version in each module to make debugging in the field some-

what easier. Do not add the version to the `cli_foo_mod` symbols. Those symbol names are inviolate and should remain with the modules for their entire usable lifetime.

I mentioned the build system as well. With the code now split into separate modules, you can easily make a build target for TEST and SHIP binaries. It is the build system that will define things such as CLI_WRITER at build time to add the module to the TEST binary. Your CI (continuous integration) system (you are using a CI system, right?!) can now pop out binaries of both types and even run the release script that tests for the presence of the correct modules in each release.

When you cannot take away the club, sometimes you can give the monkey a less-dangerous club. Putting the dangerous debug code under `#ifdef` protection, splitting the code into its own modules, and modifying the build and release system to help you ensure you do not ship the wrong thing are just some of the ways to shrink the monkey's club.

KV

Related articles on queue.acm.org

Porting with Autotools

Using tools such as Automake and Autoconf with preexisting code bases can be a major hassle.
Kode Vicious
<http://bit.ly/2BZ6tCy>

Playing for Keeps

Will security threats bring an end to general-purpose computing?
Daniel E. Geer, Verdasys
<http://bit.ly/2z3ISIE>

Security Problem Solved?

Solutions to many of our security problems already exist, so why are we still so vulnerable?
John Viega, Secure Software
<http://bit.ly/2jxA7vj>

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

Copyright held by author.

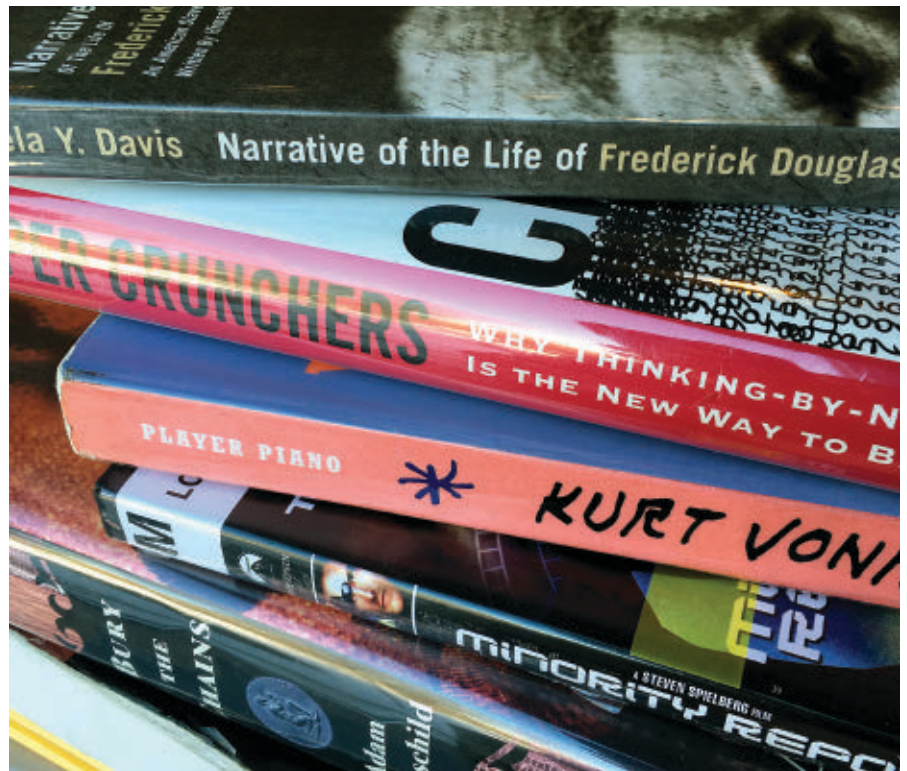
Viewpoint

Teaching Artificial Intelligence and Humanity

Considering rapidly evolving human-machine interactions.

EMERGING ANXIETIES PERTAINING to the rapid advancement and sophistication of artificial intelligence appear to be on a collision course with historic models of human exceptionality and individuality. Yet it is not just objective, technical sophistication in the development of AI that seems to cause this angst. It is also the linguistic treatment of machine “intelligence.” Headlines decry the existential threat of machines against humans in various media outlets. But what is really at stake?

Are we truly concerned that we will be surpassed in our capacities as human beings? Or is rhetorical slippage betraying age-old philosophical questions on what it really means to be human? To what degree do our shortcomings in acknowledging human dignity in all populations (regardless of skin pigmentation, linguistic system spoken, geographical location, or socioeconomic position) emerge in questions pertaining to power dynamics between humans and machines? And how might we usefully juxtapose a historic study of our past categorical taxonomies of humanity to more subtly inform our navigation of human-machine relationships? In the fall of 2017 we engaged these questions and more with first-year students at Carnegie Mellon University: 16 students from the School of Computer Science and



the Robotics Institute and 16 students from the Dietrich College of Humanities and Social Sciences. In a time of accelerating technological disruption, the next generation of leaders and innovators are ill-equipped to navigate this boundary chapter in human-machine relationships. Perhaps our students can learn from how humans have treated humans to determine viable roadmaps for this challenging mo-

ment in our economic, social, and political history, as we mindfully navigate human-machine interactions.

The ways in which machine systems influence our lives have become more explicit in recent years. A chief example that commands popular attention has been IBM’s Watson, serving as an informative bellwether for human-machine relations. Its inventors and user community place Wat-

son's clinical knowledge squarely within the social context of the medical community, ascribing agency and a capacity to *learn* to a sophisticated machine with a human name: "Nobody can read it all," Miyano said. "We feel we are a frog in the bottom of the well. Understanding cancer is beyond a human being's ability, but Watson can read, understand and learn. Why not use it?"⁶

Watson's capacity to process data rivals that of a practicing physician and, in some domains, outpaces human abilities. It is positioned as a tool that will rival human capacities in diagnostics to serve as release time for the practicing physician to dedicate more time, energy and intellectual bandwidth to patient-physician interactions. The optimized functions of the Watson apparatus have limitations but they are certainly becoming more sophisticated rapidly: "Before the computer can make real-life clinical recommendations, it must learn to understand and analyze medical information, just as it once learned to ask the right questions on 'Jeopardy!' ... The famed cancer institute [Memorial Sloan-Kettering] has signed up to be Watson's tutor, feeding it clinical information extracted from real cases and then teaching it how to make sense of the data. 'The process of pulling out two key facts from a 'Jeopardy!' clue is totally different from pulling out all the relevant information, and its relationships, from a medical case ... Sometimes there is conflicting information. People phrase things different ways.'"²

Read, Understand, Learn

IBM's Watson is personified as an independent agent in most press coverage. In contrast, at expert conferences like the "Humans, Machines and the Future of Work" conference at Rice University, AI systems like Watson are described as tools. Personification is more tightly regulated when discussed or presented to technologists who are not beholden to the mysteries of the black box, but rather its deconstruction into computational techniques. In the public domain, however, journalists ascribe personhood to the learning machine, which is not necessarily corrected by engi-

neers or physicians, by describing the machine's functions as *reading* and *learning*. Is Watson's information processing and model-building truly reading or understanding? Does such a machine *learn*? Why do we ascribe features historically associated with humanity, subjectivity, and notions of a human self to built machines? And what chapters of human interrelationships are threatened when we readily ascribe human characteristics to engineered systems?

Our society is locked in a stance of both anxiety and ambition in regard to the future of AI. We believe it is crucial that students embarking on undergraduate studies, as budding technologists, writers, policymakers, and a myriad of other future leadership roles, should be better equipped and better practiced in engaging these difficult questions. As automation will be a distinguishing feature in the next chapter of global economies, underemployment threatens the dignity of much of our human labor force. Yet as humans, most individuals would argue they are considerably more than a simple labor force driving a (albeit pervasive and powerful) global economy. An insistence on our capacity to be more than what we might produce as commodities in a market is a distinguishing feature of human dignity in the 21st century. This is a concept, however, that needs to be tested, explored and seriously considered as students prepare to enter this labor force and shape its direction for the coming generations.

**Our society
is locked in
a stance
of both anxiety
and ambition
in regard to
the future of AI.**

As humans we are readers, we create, we imagine, we strive to understand. Our individual subjectivity allows us to do more than perform specific functions. And yet, present discourse on the potential for AI is oftentimes laced with echoes of our anxieties pertaining to human dignity and its links to work or the distinctiveness of our subjectivity and agency.⁴ Will we be 'bested' by the very machines that we build? Will the next generation of technologists be equipped to consider these intended and unintended consequences for the tools they unleash? Or, for now, do we only dream quite wildly beyond technological purviews about the actual sophistication of these tools?

Reading

In the historical context of globalization, labor, human dignity, and education in the West, few rival the narrative potency offered by Frederick Douglass. In *The Narrative of the Life of Frederick Douglass, An American Slave* he writes: "Very soon after I went to live with Mr. and Mrs. Auld, she very kindly commenced to teach me the A,B,C. After I had learned this, she assisted me in learning to spell words of three or four letters. Just at this point of my progress, Mr. Auld found out what was going on, and at once forbade Mrs. Auld to instruct me further, telling her, among other things, that it was unlawful, as well as unsafe, to teach a slave to read."⁵

The power of literacy and its capacity to equip individuals with the necessary tools to dismantle exploitative and unjust systems of power are illustrated in Douglass' work. His capacity to articulate the features of a power negotiation that undermines the very core of the master-slave relationship in a post-Enlightenment era is captured in a human capacity to learn. In the context of the West and its political and social systems, literacy is an opportunity to assert agency. But human-to-human capacities to assert equality, to facilitate Douglass' ability to be 'of no value to his master,' to render him to be 'forever unfit ... to be a slave' due to his capacity to read, are not the tenets used to describe tensions between an AI machine and the tool's "master" [read programmer or

user]. In the context of Douglass' narrative, the prospect of literacy suggests the slave as worth more than his or her labor. Instead, the slave's capacity to learn, to engage in civilized discourse by joining what Benedict Anderson calls "an imagined community," suggests his equal position with other members of the society in contrast to the juridical category of slave as property.¹ It is not the same scenario with AI because the machine is not human, although they are becoming increasingly social. This is relevant because for a long period (arguably a period that we still occupy) humans have treated other humans as tools. The institution of slavery, and its cousin in European colonial systems worldwide, used humans as machines composing a labor force. These were beings who were not extended existential features like agency, subjectivity, individuality, or intelligence. As Joseph Conrad wrote in *Heart of Darkness*, the enslaved Congolese were "black shadows," they were "bundles of acute angles sat with their legs drawn up."³

Reading, however, marks agency for Douglass. Reading, in the context of an AI system, suggests anthropomorphic undertones and perhaps humanity. The hierarchical power relationships suggested in these examples are not in the service of hyperbole. The themes they introduce are also not entirely new. The historical analysis of human subjugation offers a rich tapestry regarding agency, identity, autonomy, labor, dignity and citizenry—issues at the heart of how future AI systems and humans will interrelate in our near future. *Reading* and *learning*, the very verbs ascribed today to Watson, are central to Frederick Douglass' discovery of the ways in which illiteracy in enslaved populations reinforces the hegemonic power structure of the master-slave dynamic before abolition. Through reading, Douglass asserts his freedom in deed if not in legal standing.

Understanding

Ascription of features like inconsistency, induction and emotion to machines prematurely suggests essential human characteristics upon our inventions. Yet technologists forge ahead, projecting personhood and agency,

Core human characteristics become terms for making sense of complex robotic behavior.

coupled with anxiety and uncertainty, upon machines. Even in the case of early light-seeking robots in 1950, W. Grey Walter recognized elements of *human psychology*, from free will to personality: "... the uncertainty, randomness, free will or independence so strikingly absent in most well-designed machines. The fact that only a few richly interconnected elements can provide practically infinite modes of existence suggests that there is no logical or experimental necessity to invoke more than 'number' to account for our subjective conviction of freedom of will and our objective awareness of personality in our fellow men."⁷

Core human characteristics become terms for making sense of complex robotic behavior, as if complexity is sufficient to justify giving our machines subjectivity. Today, serious legal experts are considering granting personhood to self-driving automobiles, because these AI-driven machines will be so socially integrated into our transportation infrastructure that they need to be individually liable for accidents. Notably, corporations were historically granted limited personhood to shield individual humans from responsibility and blame; personhood ascribed to AI similarly shields both corporations and engineers. Personification trades accountability with convenience for tort and liability, and further with product marketing. Watson, for instance, is named as one technology product across many use cases; but in reality, each disciplinary version of Watson is a separate, custom-made instantiation with its own silo of AI, data store, and interface.

Calendar of Events

February 5–9

WSDM 2018: The 11th ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, Co-Sponsored: ACM/SIG, Contact: Yi Chang, Email: garrychang@gmail.com

February 21–24

SIGCSE '18: The 49th ACM Technical Symposium on Computing Science Education, Baltimore, MD, Sponsored: ACM/SIG, Contact: Tiffany Barnes, Email: tiffany.barnes@gmail.com

February 24–28

CGO '18: 16th Annual IEEE/ACM International Symposium on Code Generation and Optimization, Vösendorf/Vienna, Austria, Contact: Jens Knoop, Email: knoop@complang.tuwien.ac.at

February 24–28

PPoPP '18: 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Vienna, Austria, Co-Sponsored: ACM/SIG, Contact: Andreas Krall, Email: andi@complang.tuwien.ac.at

February 25–27

FPGA '18: The 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, Sponsored: ACM/SIG, Contact: Jason Anderson, Email: janders@eecg.toronto.edu

March

March 5–8

HRI '18: ACM/IEEE International Conference on Human-Robot Interaction, Chicago, IL, Contact: Takayuki Kanda, Email: kanda@atr.jp

Sample of syllabus keywords (left column) and related materials for analysis in seminar (from Williams⁸).

Agency	Frederick Douglass: <i>Narrative of the Life of Frederick Douglass, an American Slave</i> <i>Black Mirror: Men Against Fire</i>
Self	<i>Black Mirror: Be Right Back</i> Jerrold Seigel: <i>The Idea of the Self</i>
Technology	Adam Hochschild: <i>Bury the Chains</i> Werner Herzog: <i>Lo and Behold</i>
Equality and Exploitation	Andrew McAfee: <i>The Second Machine Age</i> <i>Star Trek: The Measure of a Man</i>
Surveillance	<i>Minority Report</i> Ian Ayres: <i>Super Crunchers</i>
Labor and Digital Labor	Joseph Conrad: <i>Heart of Darkness</i> Simon Head: <i>Mindless, Why Smarter Machines are Making Dumber Humans</i>
Citizen	Kurt Vonnegut: <i>Player Piano</i> <i>Black Mirror: Hated in the Nation</i>
Narrative	Richard Powers: <i>Plowing the Dark</i> David Herman: <i>The Cambridge Companion to Narrative</i>

Learning

Rapid progress in AI/machine learning and its central role in our social, economic, and political culture signals its salience to the next generation of students entering universities. Building next-generation AI is currently a hot topic. At Carnegie Mellon, we have no trouble filling such classes. And yet, a nuanced understanding of the contributions that technologists are currently making to the world, an indication of how the next generation of computer scientists, engineers, and roboticists might shape the world that humanists and social scientists study, is not at the forefront of our undergraduates' minds. So, how might we ensure this is something they consider throughout their undergraduate career? And that, instead, societal consideration shapes their undergraduate studies from their first year onward? We propose to introduce *AI and Humanity* in the first term of the undergraduate career. Humanities students will sit in class beside their colleagues from the Robotics Institute and the School of Computer Science. They will be taught each class by a team of faculty with an intertwined pedagogical approach: a roboticist and a humanist.

Artificial Intelligence & Humanity is part of a new fleet of first-year courses called Dietrich College Grand Challenge Interdisciplinary Fresh-

man Seminars. These encourage faculty teams to propose courses that attend to historically persistent problems facing humanity, demonstrating an interdisciplinary approach to attending to these problems whose solutions continue to elude us or demonstrate boundary work that a single discipline is often ill-equipped to solve. By harnessing the methodological approaches of various disciplines to demonstrate the complexity and the range of approaches to problem solving in the academy, students are exposed to argumentative structures and efforts to juxtapose historical human-to-human relationships with future narratives of human-to-AI relations.

In Artificial Intelligence & Humanity, students will respond to historical examples of negotiations of power between individuals and communities, then develop language to describe contemporary and historical taxonomies of human-to-human and human-to-machine power relationships. Starting with a survey of narrative forms that explore human relationships that include written memoirs, dystopian television shows, documentary films and science fiction novels, students will consider the various ways in which we narrate our relationships between humans from a variety of perspectives. They will consider how these relationships might manifest in our

contemporary consideration of human relationships to machines. We take inspiration from Raymond Williams' *Keywords* and the Key Words Project (<http://www.keywords.pitt.edu>) to create a conceptual structure of core themes that will guide the semester (see the accompanying table).

In this interdisciplinary course, students will be introduced to both the historical development of AI and to the current state of the art. As we engage with core themes of power negotiations, political implications for advancing technology, and cultural response, students will use terminology from Key Words to build conceptual maps that make sense of technological advances and their societal implications. Students will develop mixed media 'futuring' assignments by semester's end that offer speculations on the future of human relationships to machines. Working in groups, they will create their own narratives, synthesizing a future ethic based on course materials and explorations. While this course is a first experiment in connecting the freshman experience to socio-technical issues relevant to all, we hope that several iterations of course refinement and deployment will yield an approach that can serve as a valuable scaffold for AI and humanity across institutions. **C**

References

1. Anderson, B. *Imagined Communities*. Verso, London, 1991.
2. Cohn, J. The robot will see you now. *Atlantic*. (Mar. 2013).
3. Conrad, J. *Heart of Darkness*. Bantam Books, NY, 1981, 26–27.
4. Deleuze, G. and Foucault, M. Intellectuals and power: A conversation between Michel Foucault and Gilles Deleuze. *L'Arc* (Mar. 4, 1972).
5. Douglass, F. *Narrative of the Life of Frederick Douglass, an American Slave*. Penguin Books, NY, 1982, 78.
6. Gaudin, S. IBM: In 5 years, Watson A.I. will be behind our every decision. *Computerworld* (Oct. 27, 2016).
7. Grey, W. An imitation of life. *Scientific American* (1950), 42–45.
8. Williams, R. *Keywords: A Vocabulary of Culture and Society*. Oxford University Press, Oxford, 1983.

Jennifer Keating (jkeating@andrew.cmu.edu) is Assistant Dean for Educational Initiatives in Dietrich College, Carnegie Mellon University.

Iilah Nourbakhsh (iliah@cs.cmu.edu) is Professor of Robotics at The Robotics Institute, Carnegie Mellon University.

Viewpoint

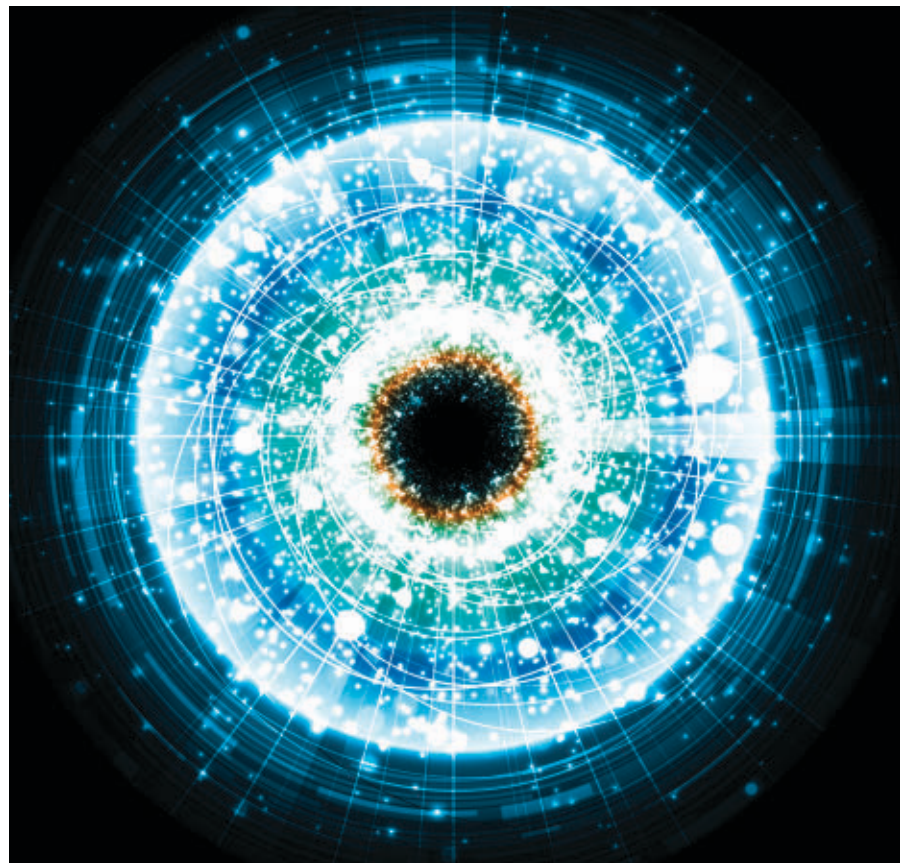
Innovation from the Edges

How innovation originates from market participants with multiple perspectives about commercial value.

SEEN FROM THE perspective of a knowledgeable observer circa 1990, the Internet would have appeared to be ill-suited for commercial life. There was no orientation toward market needs. The National Science Foundation (NSF) allowed users to develop applications that were not for sale, and most applications stressed technically advanced functionality, not user-friendliness. The network behind the scenes also did not charge prices, and seemed not to provide experience relevant to organizing a mass market for data services from many providers.

Compare those impressions with what did happen. The NSF privatized the Internet backbone, and not long thereafter the growth and deployment of the commercial Internet changed the way everyone lives, works, and plays. Most astonishing of all, the deployment of the commercial Internet brought about these changes at a fast pace. These characteristics are typically associated with only the most transformative technologies, such as the deployment of electricity or the automobile.

What lessons can be learned from this experience? Something is missing from common understanding—a view about how commercial markets shape which frontier information technology comes to users, and at what price, and in what organizational form. I developed that theme in a recent book^a and



discuss that further in this Viewpoint. As with any technology, many archetypical economic forces shaped the diffusion of the Internet. For example, the surprising start of the commercial Internet reflected well-known tensions in the transfer of technology out of universities into private hands; the rush to start businesses resembled a gold rush, and it was followed by investment typical for deploying a network good; and the bulk of investment dollars from commercial actors was oriented

to adapting technology for market needs, not with invention per se.

What was unique then? It was *innovation from the edges*, a process that arises when innovation originates from market participants with multiple perspectives about commercial value. Many perspectives brought a diversity of viewpoints to new market opportunities, reflecting a variety of opinions about how best to build businesses to address those opportunities. Many participants came from a

^a S. Greenstein. *How the Internet Became Commercial: Privatization, Innovation and the Birth of a New Network*. Princeton University Press, NJ, 2015.

ACM Transactions on Social Computing



ACM TSC seeks to publish work that covers the full spectrum of social computing including theoretical, empirical, systems, and design research contributions. TSC welcomes research employing a wide range of methods to advance the tools, techniques, understanding, and practice of social computing, particularly research that designs, implements or studies systems that mediate social interactions among users, or that develops theory or techniques for application in those systems.



For further information
or to submit your
manuscript,
visit tsc.acm.org

wide breadth of organizations, some commercial and some not. Implicit in this framework are multiple origins, that is, no single entity or commercial decision maker coordinated the Internet for an extended time with a single economic interest as motive.

This framework explains how there could be a commercial Internet that relied so heavily on markets, yet lacked an advanced plan for orchestrating the design, or a commercial firm that built and operated the entire system. It can also explain how government policies—both antitrust actions and regulatory decisions—could have encouraged or discouraged this outcome, and ended up being encouraging.

At the Outset

What policies had consequences for market participants? The Federal Communication Commission (FCC) played an important role from the beginning. It discouraged monopoly local telephone providers from interfering with the growth of new data services. Not a single U.S. phone company invested in the commercial Internet, or planned for it, but at crucial early moments they were required to do business with firms who did. This illustrates an important lesson for policy: the U.S. benefited as the location that had a much more robust set of commercial experiments than any other country.

Open organizations also played a key role. The Internet Engineering Task Force (IETF) helped govern improvement to the software protocols of the Internet. Workable standards required effort, discussion, iteration, testing, and support. To be sure, that did not happen by itself. The IETF did start with government support at DARPA and the National

Science Foundation. There is a crucial difference between mandates and encouragement, however. Aside from the FCC rules limiting monopolies, the government mandated very little. For example, DARPA and the NSF did not instruct the participants at the IETF on what they needed to do (and, to be sure, none of the participants would have taken kindly to such instruction had policy makers been willing to try). The government helped pay for the beginning, and gave the efforts its blessing, and then stepped out of decision making. The IETF's efforts were complemented by those of the World Wide Web Consortium (W3C). As is well known, Tim Berners-Lee established the W3C, and helped deploy it widely.

Most important, the IETF and W3C did not hoard the results from their efforts, nor were their lessons restricted to a small group of cognoscenti. Neither the IETF nor the W3C restricted how technology could be used, nor by whom. Instead, they helped to support a wide potential breadth of businesses after privatization. Both organizations also accumulated suggestions from many corners, scaling up their activities to adapt to the growing commercial Internet.

This openness permitted both incremental and radical change, even radical change that otherwise might have encountered roadblocks at private firms. This point is illustrated in my book with several examples from the development of the browser and the Apache webserver. Indeed, these governance norms have become an archetype themselves, and today similar behavior can be found in every open source organization, and related norms arise in many non-for profits, such as Wikipedia.

**Like any
economic activity,
the commercial
Internet possessed
a value chain.**

Support for a Surprise

Like any economic activity, the commercial Internet possessed a value chain. A value chain is a linked set of activities—typically offered by many firms—that together enable firms to sell goods and services to users. The value chain behind the commercial Internet and web did not resemble existing value chains in communications markets, which had been controlled by a small set of dominant players. Open

systems played a key role in the evolution of this new value chain.

Who took advantage of the new opportunities? The simple answer starts with “outsiders,” who specialized a wide number of different activities. At first, most commercial actions came from outsiders with distinct points of view. As an example, consider dial-up Internet service providers, many of which had been in the business of offering bulletin board services (BBS) before the commercial Internet grew. For the most part, the core of the computing and communications industry had treated BBSs as peripheral players. Eventually thousands of such ISPs provided service for the U.S. landscape.

Another key outsider came from university research. Netscape played a key role as catalyst, and this firm combined the outlook of an outsider with the financial backing of insiders. Programmers from the University of Illinois at Urbana Champaign, who had developed a browser and server software, worked with Jim Clark, an experienced entrepreneur, and his venture-capital backers, Kleiner, Perkins, Caufield, and Byers. This was just the first of many times that some commercial insiders remained open to the new outlook of outsiders and made mutually beneficial deals to develop new businesses.

Market-based Experiments

The latter part of the 1990s showed that the core architectural principles and engineering processes for operating the network could scale to a mass user base. As it turned out, the growth of investment and entrepreneurial entry persisted long after the initial rush diminished. In this case, once again, much of this commercial activity reflected economic archetypes for venture-based businesses.

In this respect, the events described illustrate another crucial lesson: Like other deployments of major technologies, the Internet was not valuable merely because it became available. The invention had to be adapted to many circumstances. Here, again, many firms from outside the core of the communications industry and computing industry perceived many of the opportunities for adaptation. For example, a large and independent ISP

There is nothing inevitable about the commercial development of technology and that also holds for innovation from the edges.

industry grew to cover many regions of the country. So too did a large number of independent contractors to provider related services, such as Web-page design and development.

Once the prototypes for the entire system were demonstrated, more commercial firms began to explore innovative applications in areas that employed frontier computing, such as back-office computing, or operations-enhancing enterprise computing in financial transactions, retailing and wholesaling, logistics, and media. The opportunities appeared large to many entrepreneurs, and while a few firms behind dot-com boom got more attention, a large number of quieter firms developed the Internet into something useful.

Established firms had to react to entrepreneurs, as many thought the entrants threatened to take leadership positions. For example, old-line firms such as IBM altered their array of services, and fundamentally altered their commercial focus. This widespread reaction and competition resembled another economic archetype, one of “creative destruction,” yielding many new services for users, and extending the deployment of the technology to many new uses. Because the opportunities extended widely, once unleashed, the phenomenon extended across virtually every sector of the economy, and in virtually every urban location in the U.S.

All this impatient investment encouraged a high tolerance for exploratory activity by commercial firms. Part of that tolerance was unsurprising in light of the scale of the new opportu-

nities and the need to adapt. In addition, many entrepreneurs came from computing, where technological races were common, and they did not find it unusual to focus on exploratory activities that stretched the functionality of software. Many such firms employed a common strategy of “getting big fast.”

The experiments had a palpable effect on perceptions about the direction of technical change in communications technologies. In 1994, hardly a boardroom in the U.S. considered the Internet a priority, and, yet, very few held that attitude in 1998. Views that had been regarded as outside the mainstream only a short time earlier were taken seriously by some of the stodgiest firms in some of the slowest moving industries. Within a few years a view that would have been considered radical a half-decade earlier had become mainstream.

Contrasting Experiences

There is nothing inevitable about the commercial development of technology, and that also holds for innovation from the edges. As already stressed, government policy can encourage or hinder it. In parallel, established firms may or may not cooperate, and the behavior of large powerful firms matters most for this latter observation. The contrast between the experience at IBM and Microsoft can illustrate both points.

Both firms profited from the rise of the commercial Internet and Web. No government had to compel either firm to sell into growing demand for their core products and services. IBM eventually sold more services as a technological intermediary and consultant. The commercial Internet and Web also increased the total number of PCs sold, and Microsoft benefited handsomely from the rising volume of software sold.

Their similarities ended there. While IBM got over its initial reluctance, Microsoft made only a temporary peace with non-proprietary standards. It was part of a long-term strategy to resist the emergence of alternative platforms, such as Netscape’s. Microsoft also used its existing contracts to encumber many of its business partners. That earned it an antitrust charge from the U.S. Department of Justice.

The ideology of the time also played

ACM Transactions on Spatial Algorithms and Systems



ACM TSAS is a new scholarly journal that publishes high-quality papers on all aspects of spatial algorithms and systems and closely related disciplines. It has a multi-disciplinary perspective spanning a large number of areas where spatial data is manipulated or visualized.

The journal is committed to the timely dissemination of research results in the area of spatial algorithms and systems.



For further information
or to submit your
manuscript,
visit tsas.acm.org

a role, and not necessarily as a positive force. For a time, Internet exceptionalism became the prevailing view. This philosophy defied the economic archetypes of business history and boldly rejected traditional approaches to building and valuing businesses. Internet entrepreneurs and their financial backers were especially vocal proponents of this view. Arguably, that encouraged innovation, because it fostered risk taking. It also wasted the opportunity, by directing growth in many wasteful projects that otherwise should have been avoided. The voices of experienced entrepreneurs and investors and analysts went unheeded. (Were these lessons heeded? The lack of reform on Wall Street suggests not.)

The dot-com crash helped put an end to the positive public reputation of Internet exceptionalism. Further exploration took on an air of renewal. The creation of value from search engines, for example, built on the interplay between advances made by a couple of precocious graduate students, Larry Page and Sergey Brin. The commercialization of their business—again—combined the views of insiders and outsiders. This happened because markets remained open to a variety of perspectives from participants with many different origins.

The advances made in wireless Internet access contained many of the same aspects. What we today call Wi-Fi drew on ideas from such a combination of insiders and outsiders. Once again, its development illustrated the lessons of government policy that encourages experimentation and does not mandate outcomes. Wi-Fi resulted from the interplay among government policy for spectrum, the design of a standardization committee, and the aspirations of many private firms.

Conclusion

How did innovation from the edges contribute to deploying the Internet? Demand for new value created a situation in which different actors who held distinct opinions about the appropriate actions could test their views. That permitted an extraordinary display of market-oriented experimentation and decentralized decision making. Collectively, it learned how to make the technology value at an extremely fast pace, and across a wide set of applica-

The dot-com crash helped put an end to the positive public reputation of Internet exceptionalism.

tions—browsers, enterprise computing, server software, wireless access, and more.

More to the point, it was far more than ever would have emerged from developing a technology inside a central decision-making process. The old telephone monopoly could not have done the same and would not have. For the same reasons no well-meaning and prescient government planner could have mandated it. That illustrates a key policy lesson: Avoid putting discretion for experiments and market investment solely in the hands of large organizations.

There was also a lesson for conventional economics, which stresses that more competitive settings lead to lower prices and induce better service than delivered by a monopoly. While that did arise in the commercial Internet—for example, more ISPs lowered prices for Internet access—the experience in this history highlights an additional observation: Competitive markets fostered a diversity of innovative viewpoints. That, in turn, supported a diversity of experiments with new commercial opportunities that otherwise would never have been conducted inside a laboratory, and on a vast scale, yielding the vibrant Internet we all use today.

As we make policies for the next generation of technologies we must not forget these crucial lessons of the recent past. Neither government alone nor solely market magic enabled good inventions to reach users and improve lives. It took a wise combination, one that avoided monopoly and government mandates, and enabled innovation from the edges. 

Shane Greenstein (sgreenstein@hbs.edu) is the Martin Marshall Professor of Business Administration and co-chair of the HBS Digital Initiative at Harvard University, Cambridge, MA, USA.

Copyright held by author.

Code 2018: Updating the ACM Code of Ethics

The ACM Code of Ethics and Professional Conduct (The Code) outlines fundamental ethical considerations to which all ACM members are expected to adhere.

The Code consists of principles for personal responsibility and guidelines for dealing with many issues computing professionals are likely to face. The Code is intended to serve as a basis for ethical decision making in the conduct of professional work, and includes considerations for individuals in leadership roles.

The ACM Committee on Professional Ethics (COPE) is updating the ACM Code of Ethics and would like your input.

The current ACM Code of Ethics was adopted in 1992, and much has changed in the 25 years since then. We are updating The Code to reflect the shifts in both technology and society.

The 2018 Code is meant to be an update of The Code, not a wholesale revision. We are particularly concerned about possible blind spots or anachronisms that may have resulted from changes in technology or the profession since 1992.

You can help define what it means to be a good computing professional by contributing to the Code 2018 project.

We have completed three drafts of suggested updates to The Code, and we need your input as we begin to work on the final draft.

Get Involved! To review the drafts and to submit your comments, visit: <https://code2018.acm.org/discuss>

Members are encouraged to take an online survey about the specific principles of the code: <https://www.acm.org/code-2018-survey>



*Committee on
Professional Ethics*

<https://ethics.acm.org>

Approaching container adoption in an already cloud-native infrastructure.

BY ANDREW LEUNG, ANDREW SPYKER, AND TIM BOZARTH

Titus: Introducing Containers to the Netflix Cloud

IN 2008, NETFLIX went all-in on cloud migration and began moving its entire internally hosted infrastructure to Amazon Web Services (AWS). Today almost all of Netflix runs on virtual machines (VMs) in AWS. A customer's catalog browsing experience, content recommendation calculations, and payments are all served from AWS.

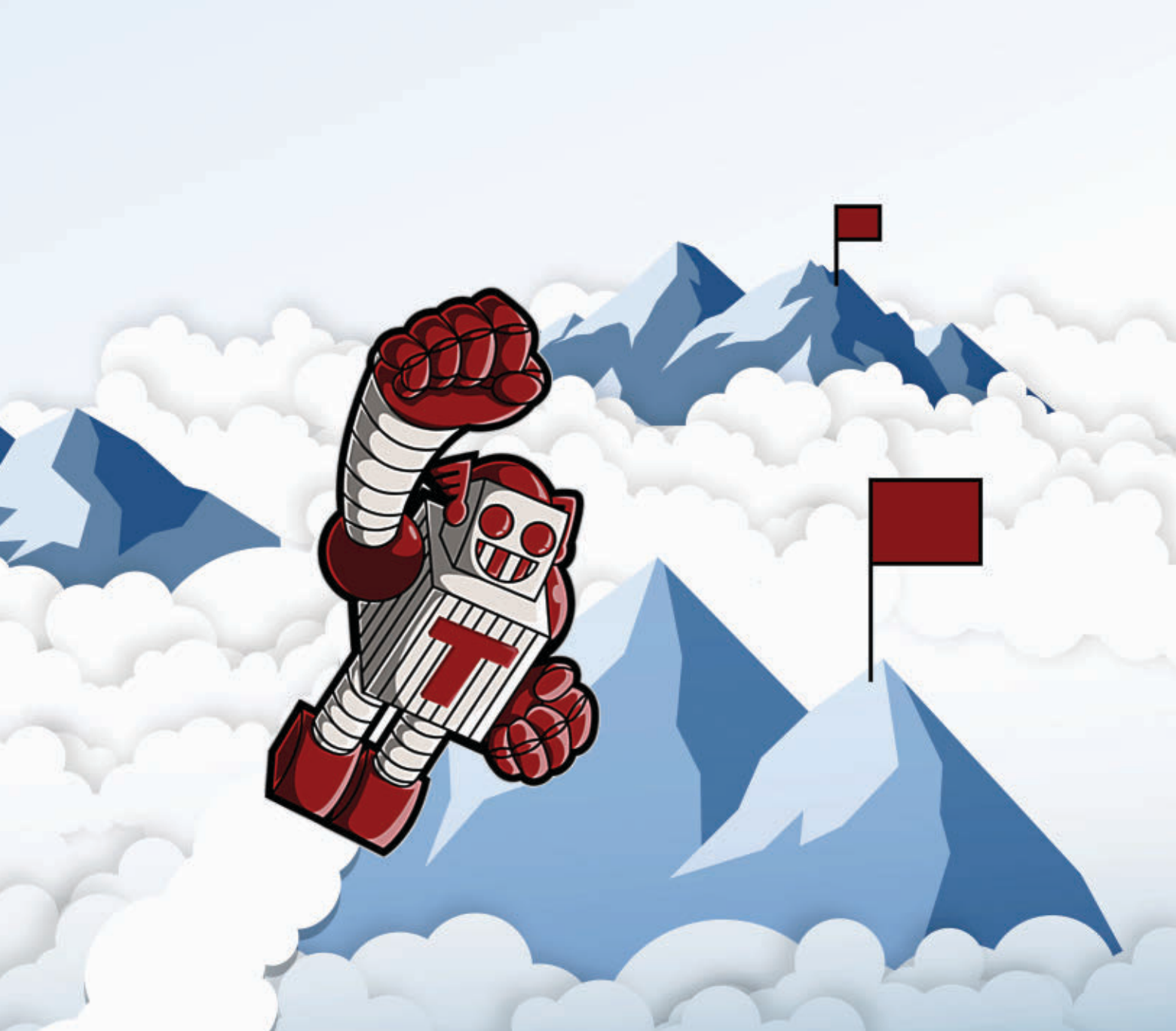
Over the years Netflix has helped craft many cloud-native patterns, such as loosely coupled microservices and immutable infrastructure that have become industry best practices. The all-in migration to the cloud has been hugely successful for Netflix. Despite already having a successful cloud-native architecture, Netflix is investing in container technology.

Container technology enables Netflix to follow many of the same patterns already employed for VMs but in a simpler, more flexible, and efficient way. Some of the factors driving this investment include:

End-to-end application packaging. Container images used for local development are identical (or at least very similar) to those that are run in production. This packaging allows developers to build and test applications more easily in a production-like environment, which improves reliability and reduces development overhead.

Flexible packaging. Netflix has historically provided a Java virtual machine (JVM)-oriented development and deployment environment, using





a common VM image that application configurations are “baked” into. For non-JVM applications, configuring this image properly can be difficult. Container images provide an easy way to build application-specific images that have only what the application needs.

A simpler cloud abstraction. Deploying Netflix applications into virtual machines requires selecting an approximately right-sized VM instance type and configuring it to run and manage the application. Many factors affect which instance type is best, including hardware (for example, CPU, memory, disk) dimensions, pricing, regional availability, and advanced feature support (for example, specialized networking or storage features). For many developers,

this is a confusing, machine-centric step that leaves opportunity for errors. Containers make this process easier by providing a more application-centric deployment that only calls for declaring the application’s requirements.

Faster and more efficient cloud resources. Containers are lightweight, which makes building and deploying them faster than with VM infrastructure. Additionally, since containers have only what a single application needs, they are smaller and can be packed more densely onto VMs, which reduces the overall infrastructure footprint.

These factors do not change the patterns or approaches to Netflix’s existing cloud-native infrastructure. In-

stead, containers improve developers’ productivity, allowing them to develop, deploy, and innovate faster. Containers are also emerging across the industry as the de facto technology to deploy and run cloud-native applications. Investing in containers ensures Netflix’s infrastructure is aligned with key industry trends.

While the value to developer productivity drove much of the company’s strategic investment, an important practical reason for investment in containers is that Netflix teams were already beginning to use them. These teams not only provided tangible evidence of how to benefit from containers, but also served to highlight the lack of internal container support.

Unique Netflix container challenges.

In many companies, container adoption happens when building new greenfield applications or as part of a larger infrastructure refactor, such as moving to the cloud or decomposing a monolithic application into microservices. Container adoption at Netflix differs because it is driven by applications that are already running on a cloud-native infrastructure. This unique environment influenced how we approached both the technology we built and how we managed internal adoption in several ways:

- ▶ Since applications were not already being refactored, it was important that they could migrate to containers without any significant changes.
- ▶ Since Netflix culture promotes bottom-up decisions, there is no mandate that teams adopt containers. As a result, we initially focused on only a few internal users and use cases that wanted to try containers and would see major benefits from adoption.
- ▶ We expect some applications to continue to run in VMs while others run in containers, so it was important to ensure seamless connectivity between them.
- ▶ Early container adoption use cases included both traditional microservices and a wide variety of batch jobs. Thus, the aim was to support both kinds of workloads.
- ▶ Since applications would be moving from a stable AWS EC2 (Elastic Compute Cloud) substrate to a new

container-management layer running on top of EC2, providing an appropriate level of reliability was critical.

Containers In an Existing Cloud Infrastructure

Netflix's unique requirements led us to develop Titus, a container-management system aimed at Netflix's cloud infrastructure. The design of Titus focuses on a few key areas:

- ▶ Allowing existing Netflix applications to run unmodified in containers,
- ▶ Enabling these applications to easily use existing Netflix and AWS cloud infrastructure and services,
- ▶ Scheduling batch and service jobs on the same pool of resources, and
- ▶ Managing cloud capacity effectively and reliability.

Titus was built as a framework on top of Apache Mesos,⁸ a cluster-management system that brokers available resources across a fleet of machines. Mesos enabled us to control the aspects we deemed important, such as scheduling and container execution, while handling details such as which machines exist and what resources are available. Additionally, Mesos was already being run at large scale at several other major companies.^{7,12,14} Other open-source container-management systems, such as Kubernetes¹⁰ and Docker Swarm,⁶ which were launched around the time Titus was developed, provided their own ways of scheduling and executing containers. Given the specific requirements noted here, we

felt we would end up diverging from their common capabilities quickly enough to limit their benefits.

Titus consists of a replicated, leader-elected scheduler called Titus Master, which handles the placement of containers onto a large pool of EC2 virtual machines called Titus Agents, which manage each container's life cycle. Zookeeper⁹ manages leader election, and Cassandra¹¹ persists the master's data. The Titus architecture is shown in Figure 1.

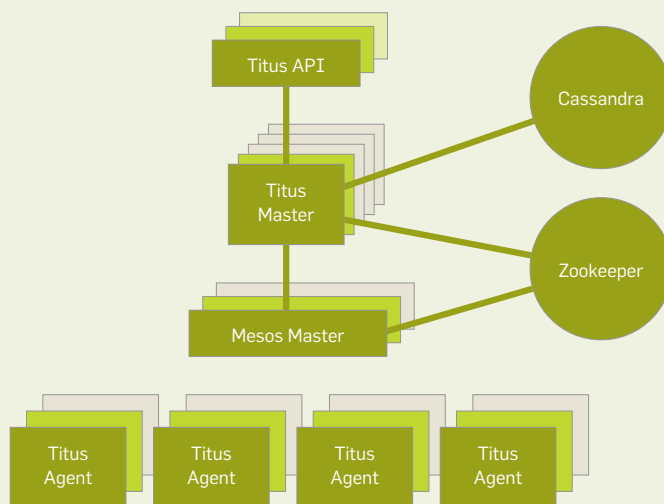
Work in Titus is described by a job specification that details what to run (for example, a container image and entry point), metadata (for example, the job's purpose and who owns it), and what resources are required to run it, such as CPU, memory, or scheduling constraints (for example, availability zone balancing or host affinity). Job specifications are submitted to the master and consist of a number of tasks that represent an individual instance of a running application. The master schedules tasks onto Titus agents that launch containers based on the task's job specification.

Designing for easy container adoption. Most Netflix microservices and batch applications are built around parts of Netflix's cloud infrastructure, AWS services, or both. The Netflix cloud infrastructure consists of a variety of systems that provide core functionality for a Netflix application running in the cloud. For example, Eureka,¹⁸ a service-discovery system, and Ribbon,²¹ an IPC library, provide the mechanism that connects services. Atlas,¹⁶ a time-series telemetry system, and Edda,¹⁷ an indexing service for cloud resources, provide tooling for monitoring and analyzing services.

Many of these systems are available as open source software.²⁰ Similarly, many Netflix applications use AWS services such as S3 (Simple Storage Service) or SQS (Simple Queue Service).

To avoid requiring the applications using these services to change in order to adopt containers, Titus integrates with many of the Netflix cloud and AWS services, allowing containerized applications to access and use them easily. Using this approach, application developers can continue to depend on these existing systems, rather than needing to adopt alternative, but similar, infra-

Figure 1. Titus architecture components.



structure. This differs from other container-management systems that either provide their own or use new, container-specific infrastructure services.⁵

Integrating with an existing cloud.

In some cases, enabling access to Netflix cloud infrastructure systems through Titus was quite simple. For example, many of the Java-based platform service clients required only that Titus set specific environment variables within the container. Doing so automatically enabled usage of the distributed configuration service,¹⁵ the real-time data pipeline system,²⁷ and others.

Other integrations required changes to the infrastructure services themselves to be able either to communicate with the Titus control plane (usually in addition to EC2) or to understand container-level data. For example, the Eureka client was updated to understand services registering from both an EC2 VM, as well as a Titus container. Similarly, the health-check polling system was changed to query Titus and provide health-check polling for containers in addition to VMs. The on-instance Atlas telemetry agent was changed to collect and emit container-level system metrics (for example, CPU and memory usage) from Titus agents. Previously, it collected only metrics for the entire host.

In addition to allowing Netflix applications to run in containers more easily, these integrations lowered the learning curve required to adopt containers within Netflix. Users and teams were able to use tools and processes they already knew, regardless of whether they were using VMs or containers. As an example, a team with existing Atlas telemetry dashboards and alerts could migrate their applications from VMs to containers, while keeping their telemetry and operations systems the same.

Integrating with the Netflix cloud infrastructure also allowed the Titus development team *not* to focus on rebuilding existing internal cloud components. In almost all cases, the effort to integrate with an existing Netflix service was far easier than implementing or introducing a new container-specific version of that service.

Rather than implement various deployment strategies in Titus, such as Red/Black or Rolling Upgrade, we



Container adoption at Netflix differs because it is driven by applications that are already running on a cloud-native infrastructure.



chose to leverage Spinnaker,²² Netflix's continuous-delivery tool. Spinnaker provides the concept of a *cloud provider*, which allows it to orchestrate application deployments across Titus, as well as EC2. In addition to providing a familiar deployment tool on Titus, the use of Spinnaker allowed the Spinnaker team, which specializes in continuous delivery, to implement the logic of how to orchestrate deployments, while the Titus development team was able to focus on container scheduling and execution.

To be sure, there are aspects of the Netflix cloud that either work differently or do not work with Titus. By integrating with existing Netflix components, however, rather than requiring that new ones be used, each of the integrations Titus provided served to lower the adoption curve incrementally for some teams and users.

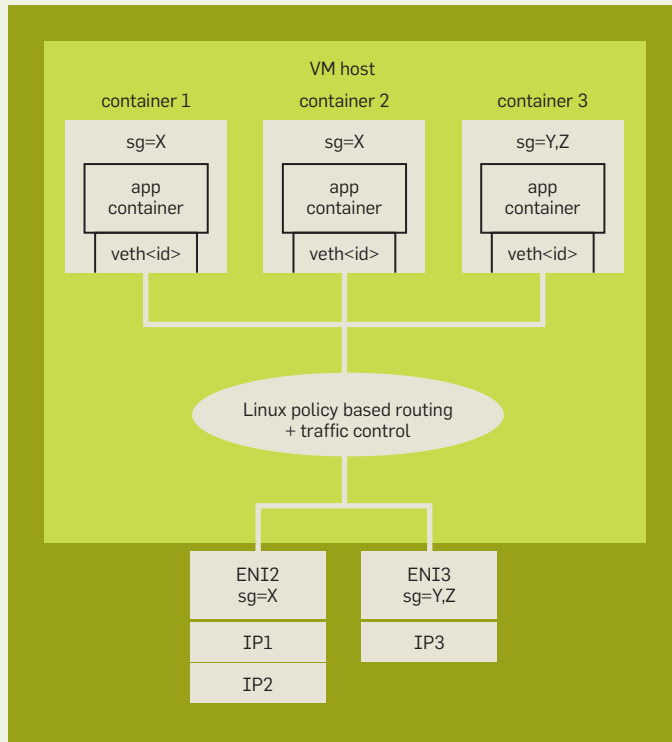
Enabling AWS integration. Another critical aspect of making container adoption easy is enabling usage of AWS services. Many Netflix applications are built around various AWS services such as S3 or SQS. Using AWS services requires the correct IAM (Identity and Access Management)³ credentials to authorize service calls.

For applications running in EC2 VMs, identity and credential information is provided via instance metadata by a metadata service⁴ that is available at a well-known IP address. This metadata service provides credentials at the granularity of an EC2 VM, which means that containerized applications on the same VM must either share the host's IAM credentials, which violate the principle of least privilege, or not use AWS services.

Titus uses a metadata service proxy that runs on each agent VM and provides containers with just their specific IAM credentials. Titus jobs declare the IAM role that is needed. When containerized applications make IAM requests to the metadata service IP, the proxy intercepts these requests via host-routing rules that redirect these requests to it.

The proxy extracts the container's IAM role from the task's configuration info that Titus provides and then uses the host's IAM *assume role* capability to acquire the specific IAM credentials and return them to the container. The IAM *assume role* allows a principal's

Figure 2. Titus container IP configuration.



IAM role (in this case, the host's) to assume the identity capabilities of another principal temporarily (in this case, the container's). This approach provides containers with only *their* IAM credentials using the same IAM roles that are used in EC2 VMs. In addition to IAM roles, the proxy provides containers with Titus instance identity information instead of EC2 identity information. Client libraries such as the Eureka client use this information.

A common network is key. An important enabler for many of the integrations was a common networking infrastructure. Seamless network communication between containerized applications and existing infrastructure removed many integration and adoption hurdles.

A common solution to container networking is to provide an overlay network that creates a separate network on top of an existing one. This approach is appealing because it decouples the two networks and does not require changing the underlying one. However, an overlay segregates the containers' networking space from the existing network and requires gateways or proxies to connect them.

Another common approach is to allocate specific ports from the host's IP to containers. While this allows the container's IP to be part of the existing network IP space, it allows containers to use only specific ports that they must know upfront, limits colocating containers that use the same ports, and exposes the host's networking policies to the container. Additionally, applications and infrastructure must handle container networking (ports) differently from how they handle VM networking (IPs). Since many Netflix systems are IP aware, but not port aware, retrofitting additional port data into the necessary systems would have been significant.

Titus provides a unique IP address to each container by connecting containers to the same AWS Virtual Private Cloud (VPC) network to which VMs are connected. Using a common VPC allows containers to share the same IP address space as VMs and use the same networking policies and features such as AWS SGs (Security Groups). This approach avoids the need to manage ports, as each container gets its own IP and full port range, and network gateways.

When launching a container that requested a routable IP address, Titus attaches an AWS ENI (Elastic Network Interface)² to the agent VM that is running the container. Attaching an ENI creates a new network interface on the VM from which multiple IP addresses can be assigned. These addresses are allocated from the same classless inter-domain routing (CIDR) range as VMs in the VPC, meaning containers and VMs can directly address each other's IPs. Allocating IPs via ENIs lets Titus avoid managing the available VPC IPs or directly modifying VPC routing tables.

When Titus is preparing to launch a new container, it creates a network namespace for it, assigns it a specific IP address from an ENI, and connects the container's network namespace to the host's using a veth (virtual Ethernet) interface. Routing rules on the host route all traffic for that IP to the veth interface, and routing rules within the network namespace configure the allocated IP for the container.

Another benefit of containers sharing the same VPC network as VMs is that they use common network security policies, such as AWS Security Groups that provide virtual firewalls.¹ Each ENI can be configured to use a set of SG firewall rules that apply to any traffic coming in or out of the interface. Titus applies a container's requested SGs to the ENI with which that container is associated, allowing enforcement of SG firewall rules to its traffic.

The number of ENIs that can be attached to a VM is limited and potentially less than the number of containers that Titus could assign to it. To use ENIs more efficiently, Titus allows containers to share the same ENI. This sharing, however, is possible only when containers use the same security group configurations, as SGs can be configured only for the entire ENI. In this case, each container would have a unique IP address, with common firewall rules being applied to their traffic. An example of sharing an ENI is shown in Figure 2. In this example, each of the three containers has a unique IP address, allocated from ENIs attached to the host. Containers 1 and 2, however, can route traffic through the same ENI because they both use only Security Group X.

Titus Master enables this sharing by treating SGs and ENIs as two-level resources so that it can schedule containers behind existing ENIs with the same SG configurations. Titus also provides guaranteed network bandwidth to each container via Linux traffic control. It sets a token bucket rate based on the bandwidth requested by the container. These networking features avoid changes to applications migrating to containers and make an application running inside a container or VM transparent to external services.

Having a common networking infrastructure eased container adoption. External services that need to connect to an application do not have to care which technology the application is using. This transparency allows existing systems to work more easily with containerized applications and makes a hybrid environment with both VMs and containers more manageable.

Supporting both batch and service workloads. Early Netflix container use cases involved both batch-processing jobs and service applications. These workloads differ in that batch jobs are meant to run to completion and can have runtimes on the order of seconds to days, while services are meant to “run forever.” Rather than managing these two different kinds of workloads with two different systems, container isolation enables these jobs to be co-located, which yields better-combined cluster utilization and reduces operational burdens.

Since these two job types have different life cycles and management needs, Titus Master separates the role of *job management* from *task placement*. Job management handles the life cycle of each job type, such as a batch job’s maximum runtime and retry policy or a service job’s scaling policy. Task placement assigns tasks to free resources in the cluster and needs to consider only the task’s required resources and scheduling constraints such as availability zone balancing.

For task placement Titus uses Fenzo,¹⁹ an extensible scheduler library for Mesos frameworks. Fenzo was built at Netflix and was already being used by an internal stream-processing system called Mantis.²⁴ Fenzo assigns tasks to resource offers presented by Mesos and supports

a variety of scheduling objectives that can be configured and extended.

Titus uses Fenzo’s bin packing in conjunction with its agent autoscaling capabilities to grow and shrink the agent pool dynamically as workload demands. Autoscaling the agent pool allows Titus to yield idle, already-purchased AWS Reserved Instances to other internal systems and limit usage of AWS’s more expensive on-demand pool. Fenzo supports the concept of a *fitness calculator*, which allows the quality of the scheduling decision to be tuned. Titus uses this feature to trade off scheduling speed and assignment quality.

While Titus Master is a monolithic scheduler, this decoupling is a useful pattern because it leverages some aspects of the two-level scheduler design.²⁵ Fenzo acts as a centralized resource allocator, while job managers allow decoupled management for different job types. This provides each job manager with a consistent strategy for task placement and agent management and needs them to focus only on the job life cycle. Other schedulers²⁶ have a similar separation of concerns, but Fenzo provides a rich API that allows job managers to support a variety of use cases and can potentially be extended to support job types with specialized needs.

Building a monolithic scheduler with separate job managers differs

from other Mesos-based systems where different kinds of jobs are managed by different Mesos frameworks. In these cases, each framework acts as a full, independent scheduler for that job type. Titus is designed to be the *only* framework on the Mesos cluster, which avoids the need for resource locking and the resource visibility issues that can occur with multiple frameworks, and allows for task placement with the full cluster state available. Avoiding these issues helps Titus Master schedule more quickly with better placement decisions.

Heterogeneous capacity management. One of the benefits of using containers through Titus is that it abstracts much of the machine-centric management that applications were doing in VMs. In many cases, users can tell Titus to “run this application” without worrying about where or on which instance type the container runs. Users still want some guarantees, however, around if or when their applications will run. These guarantees are particularly important when running applications with differing objectives and priorities. For example, a microservice would want to know it was capable of scaling its number of containers in response to increased traffic, even though a batch job may be consuming significant resources by launching thousands of tasks on the same cluster.

Additionally, applications running

Figure 3. The critical and flexible tiers.



in EC2 VMs have become accustomed to AWS's concept of Reserved Instances that guarantee VM capacity if purchased in advance. To enable a more consistent concept of capacity between VMs and containers, Titus provides the concept of *tiers* and *capacity groups*.

Titus currently provides two tiers: one that ensures Titus Agent VMs are up and ready to run containers, and one that allows the agent pool to scale up and down as workload changes, as shown in Figure 3. The chief difference between the two is the time offered to launch a container. The first tier, called the *critical tier*, is shown by the solid border in the figure. It enables Titus to launch containers immediately, without having to wait for EC2 to provision a VM. This tier optimizes around launch latency at the expense of running more VMs than the application may require at the moment.

The second tier, called the *flexible tier*, provisions only enough agent VMs to handle the current workload (though it does keep a small headroom of idle instances to avoid overly aggressive scale-up and down). Scaling the agent VMs in the flexible tier allows Titus to consume fewer resources, but it can introduce launch latency to tasks when an EC2 VM needs to be provisioned before the container can be launched. Often the critical tier is used by microservices that need their applications to scale up quickly in response to traffic changes or batch jobs with elements of human interaction—for example, when a user is expecting a real-time response from the job. The number of agents is scaled up and down as needed, shown by the dotted border in the figure.

Capacity groups are a logical concept on top of each tier that guarantee an application or set of applications some amount of dedicated capacity. For example, a microservice may want a guarantee that it will have capacity to scale to match its peak traffic volume, or a batch job may want to guarantee some amount of task throughput. Prior to capacity groups, applications on Titus were subject to starvation if other applications consumed all cluster resources (often these starvations were caused by bugs in scripts submitting jobs or by users who did not consider the amount of capacity their jobs would consume). Additionally, capac-



One of the benefits of using containers through Titus is that it abstracts much of the machine-centric management that applications were doing in VMs.



ity groups help users think about and communicate their expected capacity needs, which helps guide Titus's own capacity planning.

Combining capacity groups and tiers allows applications to make trade-offs between cost (setting aside possibly unused agent resources for an application) and reliable task execution (ensuring an application cannot be starved by another). These concepts somewhat parallel the AWS concepts of Reserved Instances and On-Demand Instances. Providing similar capacity concepts eases container adoption by allowing users to think about container capacity in a similar way to how they think about VM capacity.

Managing Container Adoption

Beginning to adopt new technology is difficult for most companies, and Netflix is no different. Early on there were competing adoption concerns: either container adoption would move too quickly and lead to scale and reliability issues as Titus matured, or container adoption would be limited to only a few use cases and not warrant investment.

Despite these concerns and the lack of internal support, a small set of teams were already adopting containers and realizing benefits. These early users provided concrete use cases where containers were solving real problems, so the initial focus was on their cases. We hypothesized that these early adopters would demonstrate the value of containers and Titus, while also allowing us to build a foundation of features that could be generalized for future use. The hope was this approach would serve to let adoption happen organically and mitigate the concerns mentioned earlier.

These early teams ran a variety of ad hoc batch jobs and made sense as initial Titus users for several reasons. First, their use cases were more tolerant of Titus's limited availability and performance provided early on; a Titus outage would not risk the Netflix customer experience. Second, these teams were already using containers because their data-processing frameworks and languages made container images an easy packaging solution. Third, many users were data scientists, and the simplified interface appealed to their desire not to manage infrastructure. Other teams were also interested in Titus but were intentionally turned away

because they were not good fits. These teams either would not see significant benefits from containers or had requirements that Titus could not easily meet at this stage.

The early users drove our focus on Netflix and AWS integrations, scheduling performance, and system availability that aided other early adopters. As we improved these aspects, we began to work on service job support. Early service adopters included polyglot applications and those where rapid development iteration was important. These users drove the scheduler enhancements described earlier, integrations commonly used by services such as the automated canary-analysis system, and better end-to-end developer experience.

Titus currently launches around 150,000 containers daily, and its agent pool consists of thousands of EC2 VMs across multiple AWS regions. As usage has grown, so has the investment in operations. This focus has improved Titus's reliability and scalability, and increased the confidence that internal teams have in it. As a result, Titus supports a continually growing variety of internal use cases. It powers services that are part of the customer's interactive streaming experience, batch jobs that drive content recommendations and purchasing decisions, and applications that aid studio and content production.

Future Focus Areas

So far, Titus has focused on the basic features and functionality that enable Netflix applications to use containers. As more use cases adopt containers and as the scale increases, the areas of development focus are expected to shift. Examples of key areas where Netflix plans on investing are:

Multi-tenancy. While current container technologies provide important process-isolation mechanisms, they do not completely eliminate noisy neighbor interference. Sharing CPU resources can lead to context-switch and cache-contention overheads,^{28,13} and shared kernel components (for example, the Network File System kernel module) are not all container aware. We plan on improving the isolation Titus agents provide at both the user-space and kernel levels.

More reliable scheduling. For both batch and service applications, there

are a number of advanced scheduler features that can improve their reliability and efficiency. For example, Titus currently does not reschedule a task once it is placed. As the agent pool changes or other tasks complete, it would be better for the master to reconsider a task's optimal placement, such as improving its balance across availability zones.

Better resource efficiency. In addition to more densely packing EC2 VMs, Titus can improve cloud usage by more intelligently using resources. For example, when capacity groups are allocated but not used, Titus could run preemptable, best-effort batch jobs on these idle resources and yield them to the reserved application when needed.

Similarly, Netflix brokers its already purchased but idle EC2 Reserved Instances among a few internal use cases.²³ Titus could make usage of these instances easier for more internal teams through a low-cost, ephemeral agent pool.

While only a fraction of Netflix's internal applications use Titus, we believe our approach has enabled Netflix to quickly adopt and benefit from containers. Though the details may be Netflix-specific, the approach of providing low-friction container adoption by integrating with existing infrastructure and working with the right early adopters can be a successful strategy for any organization looking to adopt containers.

Acknowledgments. We would like to thank Amit Joshi, Corin Dwyer, Fabio Kung, Sargun Dhillon, Tomasz Bak, and Lorin Hochstein for their helpful input on this article. 

Related articles on queue.acm.org

Borg, Omega, and Kubernetes

Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer and John Wilkes
<http://queue.acm.org/detail.cfm?id=2898444>

Cluster-level Logging of Containers with Containers

Satnam Singh
<http://queue.acm.org/detail.cfm?id=2965647>

Containers Push Toward the Mayfly Server

Chris Edwards
<https://cacm.acm.org/magazines/2016/12/210377-containers-push-toward-the-mayfly-server/fulltext>

References

1. AWS EC2 Security Groups for Linux instances; <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html>.

2. AWS Elastic Network Interfaces; http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ElasticNetworkInterfaces.html.
3. AWS Identity and Access Management; <https://aws.amazon.com/iam/>.
4. AWS Instance metadata and user data; <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>.
5. Cloud Native Compute Foundation projects; <https://www.cncf.io/projects/>.
6. Docker Swarm; <https://github.com/docker/swarm>.
7. Harris, D. Airbnb is engineering itself into a data-driven company. Gigaom; <https://gigaom.com/2013/07/29/airbnb-is-engineering-itself-into-a-data-driven-company/>.
8. Hindman, B. et al. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th Usenix Conference on Networked Systems Design and Implementation*. (2011), 295–308.
9. Hunt, P., Konar, M., Junqueira, F.P., and Reed, B. Zookeeper: Wait-free coordination for Internet-scale systems. In *Proceedings of the USENIX Annual Technical Conference*, June 2010.
10. Kubernetes; <http://kubernetes.io>.
11. Lakshman, A. and Malik, P. Cassandra—A decentralized structured storage system. In *LADIS*, Oct. 2009.
12. Lester, D. All about Apache Aurora; https://blog.twitter.com/engineering/en_us/a/2015/all-about-apache-aurora.html.
13. Leverich, J. and Kozyrakis, C. Reconciling high server utilization and sub-millisecond quality-of-service. In *Proceedings of the European Conference on Computer Systems*, (2014).
14. Mesosphere. Apple details how it rebuilt Siri on Mesos, 2015; <https://mesosphere.com/blog/apple-details-j-a-r-v-i-s-the-mesos-framework-that-runs-siri/>.
15. Netflix Archaius; <https://github.com/Netflix/archaius>.
16. Netflix Atlas; <https://github.com/Netflix/atlas>.
17. Netflix Edda; <https://github.com/Netflix/edda>.
18. Netflix Eureka; <https://github.com/Netflix/eureka>.
19. Netflix Fenzo; <https://github.com/Netflix/Fenzo>.
20. Netflix Open Source Software Center; <https://netflix.github.io/>.
21. Netflix Ribbon; <https://github.com/Netflix/ribbon>.
22. Netflix Spinnaker; <https://www.spinnaker.io/>.
23. Park, A., Denlinger, D. and Watson, C. Creating your own EC2 spot market. Netflix Technology Blog; <http://techblog.netflix.com/2015/09/creating-your-own-ec2-spot-market.html>.
24. Schmaus, B., Carey, C., Joshi, N., Mahilani, N. and Podila, S. Stream-processing with Mantis. Netflix Technology Blog; <http://techblog.netflix.com/2016/03/stream-processing-with-mantis.html>.
25. Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M. and Wilkes, J. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th European Conference on Computer Systems*, 2013, 351–364.
26. Vavilapalli, V.K. et al. Apache Hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, Article No. 5.
27. Wu, S., et al. Evolution of the Netflix Data Pipeline. Netflix Technology Blog; <https://techblog.netflix.com/2016/02/evolution-of-netflix-data-pipeline.html>.
28. Zhang, X. et al. CPI2: CPU performance isolation for shared compute clusters. In *Proceedings of the European Conference on Computer Systems*, 2013.

Andrew Leung (@anwleung) is a senior software engineer at Netflix, where he helps design, build, and operate Titus. Prior to Netflix, he worked at NetApp, EMC, and several startups on distributed file and storage systems.

Andrew Spyker (@aspyker) manages the Titus development team. His career focus has spanned functional, performance, and scalability work on middleware and infrastructure. Before helping with the cloud platform at Netflix, he worked as a lead performance engineer for IBM WebSphere software and the IBM cloud.

Tim Bozarth (@timbozarth) is a Netflix platform director focused on enabling Netflix engineers to efficiently develop and integrate their applications at scale. His career has focused on building systems to optimize for developer productivity and scalability at both Netflix and a range of startups.

Copyright held by authors/owners.
Publication rights licensed to ACM. \$15.00.

**Expert-curated guides to
the best of CS research.**

Research for Practice: Private Online Communication; Highlights in Systems Verification

THIS INSTALLMENT OF Research for Practice covers two exciting topics in modern computer systems: private communication systems, and verified systems programming.

First, **Albert Kwon** provides an overview of recent systems for secure and private communication. While messaging protocols such as Signal provide privacy guarantees, Albert's selected research papers illustrate what is possible at the cutting edge: more transparent endpoint authentication, better protection of communication metadata, and

anonymous broadcasting. These papers marry state-of-the-art cryptography with practical, privacy-preserving protocols, providing a glimpse of what we might expect from tomorrow's secure messaging systems.

Second, **James R. Wilcox** takes us on a tour of recent advances in verified systems design. It is now possible to build end-to-end verified compilers, operating systems, and distributed systems that are *provably* correct with respect to well-defined specifications, providing high assurance of well-defined, well-behaved code. Because these system components interact with low-level hardware like the instruction set architecture and external networks, each paper introduces new techniques to balance the tension between formal correctness and practical applicability. As programming language techniques advance and more of the modern computing stack continues to crystallize, expect these advances to make their way into production systems.

As always, our goal in this column is to allow our readers to become experts in the latest topics in computer science research in a weekend afternoon's worth of reading. To facilitate this process, we have provided open access to the ACM Digital Library for the relevant citations from these selections so you can enjoy these research results in full. Please enjoy!

—Peter Bailis

Peter Bailis is an assistant professor of computer science at Stanford University. His research in the Future Data Systems group (futuredata.stanford.edu) focuses on the design and implementation of next-generation data-intensive systems.



**Private Online
Communication**
By **Albert Kwon**

When we communicate online, we expect the same levels of privacy and anonymity as we do offline. Recent leaks, however, suggest that this is not the case, as large-scale surveillance threatens the privacy of our daily communication (see “NSA files decoded;”

<https://docubase.mit.edu/project/nsa-files-decoded/>, “NSA slide shows surveillance of undersea cables;” <http://wapo.st/2zFYuKQ>, and “NSA spying;” <https://www.eff.org/nsa-spying>).

Though perhaps as a result, there have also been significant efforts to protect privacy by both researchers and the open source community: Tor helps millions of users stay anonymous online (<https://www.torproject.org/>), and the Signal protocol (<https://whispersystems.org/>) used by the Signal Messaging App and WhatsApp brings end-to-end encrypted secure chats to more than a billion users already.

Still, many important challenges remain to ensure privacy of online communication. The sets of papers presented here highlights recent work that addresses some of the challenges. The first set of papers, on CONIKS and Certificate Transparency (CT), tackle mechanisms for secure distribution of public keys for end-to-end encryption. The next paper presents Vuvuzela, showing how two mutually trusting persons can communicate online without revealing anything about the content of the conversation or the metadata (for example, with whom and when one talks). The last paper discusses Riposte, a system in which users can send messages anonymously, meaning no one (not even the recipients of messages) can learn the sender of any message.

Public Key Infrastructures

Melara, M.S. et al.

CONIKS: Bringing key transparency to end users. In *Proceeding of the 24th Usenix Security Symposium*, 2013; <https://www.usenix.org/node/190975>

Laurie, B. et al.

Certificate transparency. IETF RFC 6962, 2013; <https://tools.ietf.org/html/rfc6962>;

End-to-end encryption is already prevalent in today’s Internet (for example, [https/TLS](https://tls)), but an important bootstrapping problem remains: How can you be sure you are encrypting for the right end point? Traditionally, we trust a small number of entities such as CAs certificate authorities (CAs) or PGP

(Pretty Good Privacy) key servers to maintain a valid list of public keys. Users can then query them to acquire the keys and start an encrypted communication channel. Unfortunately, as we have seen many times in practice (see “Iranian man-in-the-middle attack against Google demonstrates dangerous weakness of certificate authorities;” <http://bit.ly/1sbdGWk>) and “How secure is HTTPS today? How often is it attacked?” <http://bit.ly/1LegDNa>), these entities can be compromised and thus provide incorrect keys to the users.

CONIKS and Certificate Transparency (<http://sigops.org/sosp/sosp15/current/2015-Monterey/printable/136-hooff.pdf>) aim to remove the single point of trust and add transparency to the public-key infrastructures for end-user keys and Transport Layer Security (TLS) certificates, respectively. Though the details are different, the high-level ideas are similar: they both use transparency logs, which are the sets of public keys stored as Merkle trees. When a third party (for example, users, dedicated monitors, and so on) requests a public key from a CA or a key server, the response comes with a proof that can be verified efficiently to ensure the key is correct. Both systems are practical, requiring only a few extra kilobytes of data to verify a key; in particular, CT has been deployed by many CAs, and many popular browsers such as Chrome and Firefox already have built-in support.

Private Point-To-Point Communication

Lazar, D. et al.

Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015

Encryption can hide the content of the messages, but it does not hide potentially important metadata such as with whom and when one is talking. Vuvuzela is a recent work that hides as much metadata as possible by adding noise to the network to obfuscate users’ actions.

The system consists of a handful of Vuvuzela servers that act collectively



ALBERT KWON

Many important challenges remain to ensure privacy of online communication.





JAMES R. WILCOX

In recent decades, the research community has developed techniques that allow one to verify important properties of real systems.



as a privacy provider. Vuvuzela users send messages to other users in the system through the Vuvuzela servers. As each server routes the messages, it also adds many dummy messages (messages indistinguishable from those sent by the real users) such that no adversary can learn if two users are communicating with each other, as long as one of the servers remains honest; a key insight of the paper is using differential privacy to determine the quantity of dummy messages required to provide provable strong privacy guarantees. Vuvuzela can support millions of users with commercially available machines for SMS-style messaging, where the users can tolerate some amount of latency. To my knowledge, this paper was one of the first uses of differential privacy for private communication, which is exciting in its own right.

Anonymous Communication

Corrigan-Gibbs, H. et al.

Riposte: An anonymous messaging system handling millions of users. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*; <http://dl.acm.org/citation.cfm?id=2867658>.

Sometimes, one might also want to hide his or her identity from the recipient of the message. A whistleblower, for example, might wish to send a message either to a large group or audience or a particular end point, without revealing the identity of the sender. Riposte is an anonymous broadcasting system (think anonymous Twitter) that enables exactly that for millions of users.

Similar to Vuvuzela, Riposte uses a small number of servers, one of which needs to be honest to guarantee anonymity. To send a message, a user splits his or her message into multiple shares, each of which is given to one of the servers. Each server then stores each share in a database. After a large number of users submit their messages, the servers come together to reveal all messages simultaneously without revealing the senders of the messages to anyone. The system can support millions of Tweet-length messages per day and is a great example of how theory meets practice: the system has a formal proof of security, a prototype implementation, and evaluation.

Final Thoughts

As the world becomes more connected, the importance of private communication will continue to grow. The papers presented here are only a few examples of recent works on the topic. Many other interesting papers have been written about private communication. Pung (<https://www.usenix.org/conference/osdi16/technical-sessions/presentation/angel>), for example, is another private point-to-point communication system that provides privacy under an even stronger threat model at the cost of latency. Other anonymity networks such as Dissent (<https://www.usenix.org/node/170846>) and Riffle (<https://dspace.mit.edu/handle/1721.1/99859>) provide anonymity guarantees similar to Riposte but with different trade-offs.

Many important challenges remain, however, to realize private communication for everyone. To list just a few: How can we scale private communication to billions of users? How can we hold users accountable without sacrificing their privacy and anonymity? How do we make privacy user friendly? Without a doubt, many more interesting works will come in the near future.



Highlights In Systems Verification

By James R. Wilcox

Humanity now relies on software in all aspects of life, including safety-critical applications. Programmers use a spectrum of techniques to ferret out bugs, most commonly testing or static analysis. At the most rigorous end of this spectrum is formal verification, which for decades has sought to guarantee the absence of bugs using mathematical proof.

In recent decades, the research community has developed techniques that allow one to verify important properties of real systems. When reviewing this work, it is important to consider not only the guarantees each system makes, but also their assumptions; these assumptions are known as the *trusted computing base*, or TCB.

The remainder of this article highlights three applications of verification techniques to pervasive systems infrastructure: compilers, operating systems, and distributed systems. These projects point to a future where

practical systems can be built from existing verified components, eliminating entire classes of bugs—from the hardware up to the application logic.

Verified Compilers: CompCert

Leroy, X.

Formal verification of a realistic compiler.

Communications of the ACM 52, 7 (July 2009), 107–115; <https://cacm.acm.org/magazines/2009/7/32099-formal-verification-of-a-realistic-compiler/abstract>

Compiler bugs are infectious: A buggy compiler can make otherwise correct source programs misbehave at runtime. This is concerning for any programmer, but especially so if the programmer wants to reason at the source level or use source-level program analysis. Any analysis results are at risk of being invalidated by the compiler's disease.

CompCert is a C compiler that has been formally proven never to miscompile source programs. More precisely, CompCert is guaranteed to produce assembly code that is equivalent to the C source program. CompCert is programmed and proved using the Coq proof assistant.

Like all verified systems, CompCert trusts certain other pieces of software to be correct. The TCB of a verified system generally includes tools used to carry out the verification, the specification, and the *shim*, or glue code, used to connect the system to the rest of the world. CompCert's TCB contains tools such as Coq itself, the OCaml compiler and runtime, and the operating system; the specification, including the semantics of both C and the target assembly language; and its shim, which is an unverified OCaml program responsible for reading files from disk and so on.

Verified Operating Systems: seL4

Klein, G., et al.

seL4: Formal verification of an operating-system kernel. *Communications of the ACM* 53, 6 (June 2010), 107–115; <http://dl.acm.org/citation.cfm?id=1743574>

Operating-systems bugs, like compiler bugs, may cause a correct program to misbehave. Even worse, the OS strain of contagious bugs can result in unintended interaction among processes. Such interaction can lead to security holes, such as leaks of sensitive data across process boundaries.

seL4 is an operating-system kernel that is verified for full functional correctness. More precisely, seL4 is shown to refine an abstract specification of its behavior. This refinement guarantees, among other things, that no system calls ever panic unexpectedly, loop infinitely, or return wrong results. These guarantees are sufficient to establish security properties such as access control and process isolation.

The refinement proof, done in the Isabelle/HOL proof assistant, first shows that the C implementation refines an executable specification written in Haskell; the Haskell specification is then shown to refine the abstract specification.

seL4's TCB includes Isabelle/HOL itself, the C compiler, the hardware, the abstract specification, and the shim, which consists of several hundred lines of handwritten assembly.

Verified Distributed Systems: Verdi

Wilcox, J.R., et al.

Verdi: A framework for implementing and formally verifying distributed systems.

In *Proceedings of the Conference on Programming Language Design and Implementation*, 2015; <http://homes.cs.washington.edu/~mernst/pubs/verify-distsystem-pldi2015-abstract.html>

At least compiler and operating-system bugs are localized on a particular node. In contrast, avoiding distributed-systems bugs requires reasoning about the interaction between nodes. Furthermore, distributed systems must tolerate failure of the underlying hardware.

Verdi supports reasoning about distributed systems by modeling the network using *network semantics*, which formally capture the potential faults the nodes might experience, including dropped messages, crashing machines, and so on. Verdi employs verified systems transformers (VSTs), which can, for example, add fault tolerance to existing systems. Verdi has been used to verify the Raft consensus protocol as a VST from a single node to a replicated system.

Verdi trusts Coq itself, the OCaml compiler and runtime, the fact that the network obeys the fault model, and its shim, which is an unverified OCaml program responsible for low-level network and disk access.

Future Directions

The research community has now verified many pieces of common infrastructure. Going forward, how do we connect these pieces to build larger verified applications?

As a simple example, imagine combining verified systems from the domains highlighted here to build a verified replicated key-value store. Such a system would use Raft for replication; be compiled with a verified compiler; and run on a verified operating system. Today, it is not clear how to execute such a plan for several reasons.

First, the systems are written in different proof assistants—CompCert and Verdi in Coq, seL4 in Isabelle/HOL—so it is not directly possible to reason about their composition.

Next, it is likely that the systems make subtly incompatible assumptions about each other or their shared environments (for example, seL4 may use a feature of C that CompCert does not support). In a similar vein, the correctness theorems of the systems are not designed to work together logically; for example, the assumptions Verdi makes about the operating system are unlikely to be exactly what seL4 proves.

Finally, many techniques require re-implementing the system from scratch in a way that supports verification, but this is impractical in a world of large legacy systems.

We need techniques to build larger verified systems from verified components. One bright spot on this horizon is the recent DeepSpec project, which seeks to connect verification projects across many abstraction layers. Future work should seek to integrate verified systems into a library of reliable components that can be snapped together to build bug-free applications. The existence of such a library will also lower the barrier to entry for verifying systems, eventually leading to a world where it is no more expensive to verify a system than to test it thoroughly.

Albert Kwon is a Ph.D. candidate in the EECS department at MIT, where he has worked on oblivious RAMs, public key infrastructure, cryptocurrencies, anonymous reputation, and anonymous communication networks.

James R. Wilcox is a Ph.D. student at the University of Washington in the Programming Languages and Software Engineering lab (<http://uwplse.org/>).

Copyright held by authors/owners.
Publication rights licensed to ACM. \$15.00.

**Try to see things from
a manager's perspective.**

BY KATE MATSUDAIRA

Views from the Top

I CAN REMEMBER the very first software project I worked on. Back then most programming was for shrink-wrapped software that would spend years in development (since you released only every few years and had long dev cycles because patching bugs was so costly).

For two years I worked on a project, and when it finally shipped, I can remember our VP talking about the launch. I had never had much exposure to him (I was new, a grad straight out of school), but I remember his speech about the launch clearly. He talked about some of the key features and mentioned a few of the people involved.

At the time, I had the impression he was out of touch; the people he recognized were not the ones who had contributed the most code, and the features he called

out were important but not the ones that had been the major engineering challenges. I remember thinking, *"How can he not know what is going on in his team?"*

Of course, now, almost 20 years later, my perspective is quite different.

I have had the opportunity to manage very large teams; including some even bigger than the 400-person organization I was part of during that first project of my career. Now it makes perfect sense to me why he might not have known the biggest challenges or top contributors for a specific project.

The view at the top is different. And having been on both sides of the org chart, I have a new perspective.

The lessons here are ones I wish someone had shared with me in my moments of frustration with upper management earlier in my career.

Lesson 1. There Are Only a Few Levers to Effect Change

You know that a good leader empowers his or her people. It is the leader's job





to guide them, but also to trust them. This means allowing them to make mistakes. Frequently there are no right answers, and sometimes managers can go down the wrong path.

When things are not going right, these senior leaders have a limited number of options to make a change. Long term, it is not effective to step in and micromanage their direct reports, or even worse, the people on their direct reports' teams. This is not scalable, and it is expected that these experienced people should not need that level of management and direction.

Instead, leaders look for ways to get high-performing, trusted managers in a position to help them reach their goals. Generally, this is done in three ways:

- *Start and stop projects.* If a project isn't going well, a leader can cancel it and reinvest the resources elsewhere. If something isn't working, the leader can staff a new project to fix it.

- *Reorg.* This is probably one of the

most painful experiences for members of a team. It can be so upsetting to have your manager or your manager's manager move out of your chain of command, especially if you have worked to build strong relationships with them. In large organizations, however, changing the structure of a team is one of the best ways for leaders to improve alignment and strategically place their top people in positions to help them achieve better results.

- *Hire, or fire, the leadership team.* If someone isn't performing, or the team isn't moving in the right direction, a highly effective course correction is to bring in fresh energy. Of course, this is more difficult to do because ...

Lesson 2. The More People Under a Manager, the More Challenging It Is to Judge Their Effectiveness

One of my favorite questions to ask is how long it takes to tell if a VP is mediocre or great. The answer can be quite challenging to determine because a

lot of a leader's success (or failure) can be attributed to his or her team, not to the leader.

If you have strong managers under you, then it is easy to ride on their coat-tails. They make sure things are moving in the right direction and that good things are happening. Conversely, if you have poor performers, it can take a while to coach them or manage them out of your organization. The deeper the hierarchy, the more levels of indirection there are. Judging a VP isn't like judging a software engineer where you can at least observe his or her output and contributions directly.

Of course, the signs of a bad leader are not always immediately obvious—delivery on substantial projects often takes months or years, and attrition/retention tends to be a lagging indicator.

This is why bad leadership can be in place for years before changes are made. It can actually take that long to prove it is *that* person, as opposed to

other external factors outside of their control, causing failure to occur.

Lesson 3. Interviewing Senior Leaders Is Difficult to Do

Another observation I have seen play out is that it is very difficult to hire senior leadership (and because of Lesson 2, it can take a while to know if you did it right or made a mistake).

There are plenty of pitfalls in conducting job interviews, but the task becomes more challenging with executive leadership because there isn't a set of skills that is easy to test. How do you test influence? Sure, you can proxy it with a set of questions, but spending a few hours with a candidate does not always indicate accurately if he or she will be successful in the role.

That is why many companies focus on the candidate's experience and track record. Personal references and endorsements can also play a large part.

Perhaps the biggest reason this is hard, though, is that leading a group of people effectively is dependent on so many factors: the team culture, the organizational goals, and, of course, the individual personalities. What worked really well for one person in one environment doesn't always translate to a new place. That is why adaptability and flexibility are important traits to look for during the hiring process, not just past successes.

Lesson 4. Split and Delegate

When you move from being an individual contributor to a manager, you have to deal with the challenge of managing work. It becomes your responsibility to report on progress and handle status. In a small software team, this is easy: you just show up to stand-ups, collect status email messages, or create a lightweight way to poll your team.

As your org gets bigger, however, it becomes too much for you as one person to keep everything in your head. You cannot go to all of the team meetings. You cannot be present for every decision. And you have to learn to trust your leadership and delegate responsibility.

This is a good thing overall—by sharing the responsibility, you give others the chance to lead and you allow your team to grow. It can be a difficult transition, however, if you are used to being in control. It is also a place where



One of my favorite questions to ask is how long it takes to tell if a VP is mediocre or great. The answer can be challenging to determine because a lot of a leader's success (or failure) can be attributed to his or her team, not to the leader.



things can go really wrong.

If you do not have a good way of verifying details, or diving deep into areas, too much abstraction can result in unforeseen problems (which are the worst kind). To avoid this, you have to figure out checks and balances—how can you get enough oversight to have high confidence in the work being delegated, without micro-managing every detail yourself?

The most effective strategy I have seen in these circumstances is to set up regular reviews with team leadership to surface issues and help you stay involved with the day-to-day processes of the team.

Lesson 5. Be the Beacon of Hope

While it is true that misery loves company, no one loves working for a leader who doesn't portray confidence in the team's trajectory and success. People want to be inspired, and as their leader, it is your job to give them the motivation and vision to perform.

This means that even when things are bad, or you feel frustrated, you do not let it show. You need to be the person who is positive and who helps motivate people to do their best. If you don't, then who will?

Conclusion

Leadership is difficult. None of us comes to work to do a bad job, and there are always ways we can be better. So, when you have a leader who isn't meeting your expectations, maybe try reframing the situation and looking at things a little differently from the top down. 

Related articles on queue.acm.org

A Conversation with Joel Spolsky

<http://queue.acm.org/detail.cfm?id=1281887>

People and Process

James Champy

<http://queue.acm.org/detail.cfm?id=1122687>

Nine Things I Didn't Know I Would Learn Being an Engineer Manager

Kate Matsudaira

<http://queue.acm.org/detail.cfm?id=2935693>

Kate Matsudaira (katemats.com) is the founder of her own company, Popforms. Previously she worked at Microsoft and Amazon as well as startups including Decide, Moz, and Delve Networks.

Copyright held by author.
Publication rights licensed to ACM. \$15.00.

ACM Welcomes the Colleges and Universities Participating in ACM's Academic Department Membership Program

ACM now offers an Academic Department Membership option, which allows universities and colleges to provide ACM Professional Membership to their faculty at a greatly reduced collective cost.

The following institutions currently participate in ACM's Academic Department Membership program:

- Appalachian State University
- Armstrong State University
- Ball State University
- Berea College
- Bryant University
- Calvin College
- Colgate University
- Colorado School of Mines
- Edgewood College
- Franklin University
- Georgia Institute of Technology
- Governors State University
- Harding University
- Hofstra University
- Howard Payne University
- Indiana University Bloomington
- Mount Holyoke College
- Northeastern University
- Ohio State University
- Old Dominion University
- Pacific Lutheran University
- Pennsylvania State University
- Regis University
- Roosevelt University
- Rutgers University
- Saint Louis University
- San José State University
- Shippensburg University
- St. John's University
- Trine University
- Trinity University
- Union College
- Union University
- University of California, Riverside
- University of Colorado Denver
- University of Connecticut
- University of Illinois at Chicago
- University of Jamestown
- University of Memphis
- University of Nebraska at Kearney
- University of Nebraska Omaha
- University of North Dakota
- University of Puget Sound
- University of the Fraser Valley
- University of Wyoming
- Virginia Commonwealth University
- Wake Forest University
- Wayne State University
- Western New England University
- Worcester State University

Through this program, each faculty member receives all the benefits of individual professional membership, including *Communications of the ACM*, member rates to attend ACM Special Interest Group conferences, member subscription rates to ACM journals, and much more.

DOI:10.1145/3173550

Digital technology determines how (and even whether) people work as much as it determines how information produces economic activity.

BY JOHN ZYSMAN AND MARTIN KENNEY

The Next Phase in the Digital Revolution: Intelligent Tools, Platforms, Growth, Employment

DIGITAL PLATFORMS IN the computing “cloud” are fundamental features of the digital revolution, entangled with what we term “intelligent tools.” An abundance of computing power enabling generation and analysis of data on a scale never before imagined permits the reorganization/transformation of services and manufacturing. Here, we expand two central issues raised in our 2016 article “The Rise of the Platform Economy.”¹³ First, will the increased

movement of work to digital platforms provide real and rising incomes with reasonable levels of equality? The productivity possibilities of the digital era are just coming into view. The consequences will be a matter of policy and corporate strategy. Much will depend on how intelligent tools, including big data analytics, artificial intelligence, robotics, and sensors will coalesce into systems that appear to be nearly autonomous. The goal of firms could be to simply displace work and remove human intelligence from work tasks. Alternatively, it is possible for intelligent tools to help augment intelligence and capabilities, supporting rather than displacing workforce abilities. Moreover, as communities, is it possible to choose the kind of society that will result from the digital “platform economy.” Digital technology does not, in and of itself, dictate a single answer. The increasing diffusion of intelligent tools has already exposed tension between public governance and private governance of platforms. The significance is that a platform’s operation sets the rules and parameters of participant action. Digital platforms are regulatory structures and, thus, governance systems. Policy cannot just adapt to the emergence of the digital economy and society. Policy choices are indeed part of the technological trajectories themselves.

The Basics

It is not necessary to review the digital technologies themselves. The goal is rather to explore their economic and

» key insights

- **Social and political choices determine, in part, whether deployment of intelligent tools and platforms will augment human skills or replace humans as workers.**
- **Digital platforms are regulatory and governance structures that set the rules and parameters for social and economic activity.**
- **Intelligent tools simultaneously replace, transform, and create work.**



Autonomous robot (Betty) developed at the University of Birmingham for work as an office monitor.

social implications. This phase of the digital era rests on cloud computing facilitated by the increasing abundance of inexpensive computational power, storage, and transmission resources. Gradually, but inexorably, the exponential increase in computing capabilities, noted in popular media through reference to Moore's Law and

the consequences of doubling processing power every two years, and with data storage on a roughly similar trajectory, has changed the game, even as these dynamics continue their rate of change. Lifting constraints opened the current digital era, as characterized by platforms, big data, algorithmic power, and intelligent tools.

Consider platforms. Digital platforms, which we define later, are digital algorithms and software structures that run in the cloud and operate on data. The platform story is closely related to the digital transformation of services and, more broadly, manufacturing as well. Rule-based information and communication technology

(ICT) applied to service activities has initiated an algorithmic revolution. As Zysman²⁷ argued in 2006, service activities themselves are changed when they can be converted into formalizable, codifiable, computable processes with clearly defined rules for their execution. Searching for fresh language to describe a complex process, Zysman²⁷ labeled this change the “algorithmic service transformation facilitated by ICT tools,” describing it, “Services were once seen as a sinkhole of the economy, immune to significant technological or organizationally driven productivity increases. Now the IT enabled reorganization of services, and business processes more generally, has become a source of dynamism in the economy.”

Consider how the physical cranes used in ports are often sold in a bundle with port management services, and sensor-enabled farm equipment is sold bundled with soil- and plant-management services. Here, the things are embedded in services, increasing the value of both the equipment and the services to the customer.²⁰

Today’s digital platforms consist of software processing data in the cloud. As Fortran and Unix pioneer Stuart Feldman explained to us in a recent conversation, a computer science definition would be “that platforms provide a set of shared techniques, technologies, and interfaces to a broad set of users who can build what they want on a stable substrate.” As conventionally used, the term “platforms” refers to multi-sided digital frameworks that shape and intermediate the rules participants follow to interact with one another.^{10,20} Platform power is generated through direct and indirect network effects that can result in winner-take-all dynamics, conferring enormous power to the platform owner. Platforms are thus algorithm-enabled “cyberplaces” where constituents can act, interact, and transact.

These actions are highly diverse, whether categorized by market, social function, or technical character. Each platform involves its own diverse computational and market issues and questions. Goods platforms from Alibaba, Amazon, and eBay link buyers to sellers and raise questions involving, say, the power of the underlying platform

in the supply chain.¹⁴ Service platforms, in the form of labor-market-exchange platforms (such as TaskRabbit, Uber, and Upwork), connect buyers and sellers of people-delivered services, raising potential labor market conflicts, while forcing the rethinking of traditional labor market regulations. That is, there are B-to-C platforms, sharing platforms that are often C-to-C, and indeed B-to-B, including IoT arrangements, as well as platforms for Industrie 4.0.

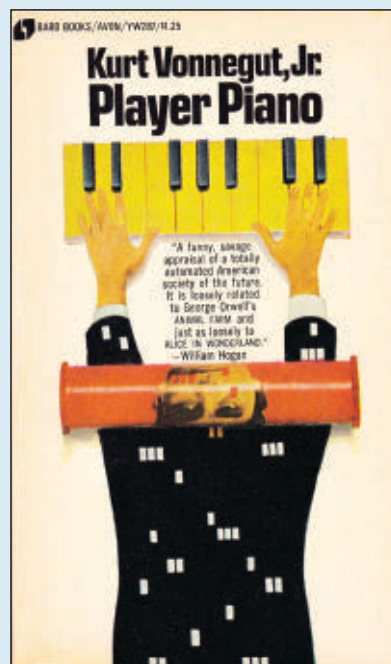
The conundrums raised are diverse and particular to each platform and industry. For example, taxis that are publicly regulated cannot discriminate among potential customers, but can Uber drivers who are indeed merely contractors discriminate against potential customers? Hotels must obey land-use rules and not discriminate among potential guests, but what about Airbnb providers? And who should enforce anti-discrimination laws and regulations—private contractors, platform owners, or the government? Who should be responsible for inspecting the algorithms driving business operations and performance? Who should have access to and control over the data private firms collect as part of their business operations and for what purpose? In terms of in-

dustrial production, the Internet of Things, a polyglot category of objects linked through cyber connections, raises even more questions about industrial standards, rules, and ownership of machine-generated data. Will standards-setting bodies set and control the industrial standards on production platforms? Will private firms create and secure adoption of de facto standards that control these interfaces? Such decisions already profoundly affect competition among producers of industrial equipment. Finally, since all such IoT-related machines are constantly producing data, who should own or have legal access to it? The market structure and relative balance of power among, say, Cisco, General Electric, Google, Huawei, John Deere, Komatsu, Siemens, small and mid-size firms, and everyone else will turn on the answers.

Cloud computing provides the computational architecture and structure for an array of interactions.¹⁵ The consequences for the user, not the “how” of cloud computing for the provider, are our focus here. Providing “computing clouds” favors scale. Scale favors players with the most demanding data processing needs and capabilities. Indeed, cloud architectures first emerged as companies like Amazon, Google, IBM, and Microsoft provided for their own computer needs, then sold excess computing capacity and services in a variety of packages. Cloud computing has matured to deliver computing services—data storage, computation, and networking—to users at the time and location and in the quantity they wish to consume, with costs based solely on resources used. Powerful computing resources can now be assembled, orchestrated, and deployed as needed. For those purchasing cloud computing as a service, the data center is no longer a capital cost, it is now simply a *variable* operating cost. This makes it possible to create, experiment with, and launch platforms at dramatically lower cost. Start-up costs are reduced, and the costs of expansion of computing resources can be managed “as needed.”

In more formal terms, cloud computing expands the availability of computing while lowering the cost of access to computing resources, sometimes to where it can be paid with an

Kurt Vonnegut’s 1952 novel *Player Piano* on the “privilege” of work in an automated economy.



individual's credit card, depending what one wants to do. This process eases access to inexpensive elastic computing resources and scaling for startups and experimentation within larger companies. The chief information officer is thus no longer a choke point for access to computing resources. One might say the cloud reduces the importance of the cost of computing when calculating the cost of starting a firm or experimenting with a new application. Organized effectively, it can speed application development and deployment. In effect, value moves up the value chain, from provision of basic computing infrastructure to creation and deployment of applications.

What sort of world will we build with platforms, data, and intelligent tools? How will value be created, and who will capture it? The pioneers of the digital age, including Robert Noyce at Intel, Bill Gates at Microsoft, and Steve Jobs at Apple, thought they were creating a world of possibility and opportunity and indeed unleashed a new way to interact with the world. Even earlier there were skeptics. For example, Kurt Vonnegut's 1952 science fiction novel *Player Piano*, based on computing machines using electronic tubes, not integrated circuits, reads like the dystopian literature seen in today's academic and popular press.^{4,5} In the world Vonnegut envisioned, work was a privilege, and, except for a privileged few who ran the system, jobs for the masses consisted of Works Progress Administration-like infrastructure repair and the military.

What kind of future will result from intelligent tools? Some part of the answer begins with these questions: What happens to productivity? How quickly will changes in jobs and work take place? What sort of jobs will be created and for whom? How are labor markets being reorganized? And who wins (and loses) and captures whatever gains there might be?

Productivity Debate

Since the mid-19th century, basic standards of living have been transformed, and the productivity of advanced economies has risen dramatically. A core debate concerns whether that historic run is continuing. ICT is profoundly transforming our lives. And yet economist Robert J. Gordon has argued that



Deployment of technology is as crucial to productivity as technology itself.



the basic changes in transport, housing, medicine, and the like that took place from 1870 to 1970 were even more profound for productivity and standards of living.¹¹

Productivity, however formally defined and measured, matters, since, at its core, it represents an organization's increased ability to generate goods and services from a given endowment of productive resources. We are collectively richer not just because of savings and investment, though they are essential, but because of sustained innovation affecting what we do and how we do it. Gordon and others have said that ICT, despite the hype, actually *has not* resulted in sustained productivity increase in the past decades.¹¹ Setting aside the observation that much of the value of ICT in the consumer marketplace, from search to social media, is provided free, in exchange for users being subject to advertising, and consequently the benefit may not be measured effectively. There have been debates over measurement before.⁶ Let us accept for the moment Gordon's finding that the drop-off in the rate of productivity increase since 1972 is real. His conclusion that after 2007 labor productivity grew at no more than 1.3% annually is sobering, particularly as this productivity growth was significantly slower than the 2.0% growth from 1891 to 2007. A core question is not why growth slowed but why and what ICT might have to do with it.

Transformative technologies, those involving a broad swath of activities as they are introduced, are believed by economists from Joseph Schumpeter²³ to Carlota Perez²¹ to drive rapid growth and productivity. The historic roles of steam engines, railroads, and electricity demonstrate the effect of these powerful general-purpose technologies.¹² The core argument by Gordon and others is that ICT, beginning with the microelectronics revolution, has not had the impact of earlier transformative technologies. That contention has two components: the proposition that ICT has had only limited scope in the economy, to, one might say, entertainment and the acceleration of financial transactions; and that the technology wave has passed, so the effects are complete and proved to be limited. Both assertions are open to debate, if not simply mistaken.

ICT is certainly a powerful general-purpose technology that laid the groundwork for Schumpeterian transformations in production organization, product design, and business models that is today recasting a significant portion of the world economy. The early phases of the ICT revolution principally affected services that, at their core, are about information, involving communications, finance, media, and insurance.²⁸ ATMs substituted for tellers in one existing business model, and while high-frequency trading on Wall Street radically changed competition in the financial sector, the basic business models were unchanged. In other sectors, established business models are indeed being overturned. The offshoring of service work to locations like India and the Philippines was possible only because content was digitized. When media content was converted to digital formats and easily distributed, traditional business models were upended. More important, in the early Internet phase of the digital revolution, ICT-enabled services, as mentioned earlier, began to be extended to “everything,” and the related business models often changed character. Examples of such change abound, some well known, others less discussed in the business press and scholarly research. For example, airplane engines, and indeed truck tires, can be sold as services with charges related to use. Finally, in 2018 the impact of online purchasing, as well as other forms of e-commerce, are only beginning to be felt in retail, as brick-and-mortar stores are closing at an astonishing rate. This will likely have a positive effect on national productivity but a negative effect on employment.

The platform phase is the latest step in this unfolding story of the deployment of ICT technologies. For the moment, consider platforms. Platforms, digital and multi-sided, provide new ways for users, who could not previously reach each other and thus could not previously form a market, to interact. The Internet of Things, Internet of Everything, and Industrial Internet amount to new ways for sensor-enabled objects to be controlled and interact through platforms. The platforms themselves facilitate aggregation and analysis of data with the intent of controlling systems and actions. We are



Capturing the promise of the technology is as much a political problem as it is a narrowly economic constraint.



entering a world that will increasingly be organized through the interplay of algorithms and data. It will be a data analysis-based economy and society where observation and interpretation of our individual behavior and optimization of our physical systems will be based on computation.

The breadth and dimensions of the effects of platforms, sensor-based system, and data analytics are breathtaking. In the prosaic world of industry, Cisco, General Electric, IBM, Huawei, and Siemens, through marketing and business strategies, highlight industrial applications, from energy management to pipelines to aircraft management. For example, General Electric says its goal is to integrate ICT and data to provide solutions across industries, including manufacturing, aviation, transportation, power generation, health care, and energy.

The provocative German discussion of Industrie 4.0 (https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf) envisions how data capture and analytics will reform and reorganize manufacturing and supply chains. German global competitive advantage in manufacturing depends on skilled labor and specialized small- and mid-size firms.¹² The question the original Industrie 4.0 study posed in Germany and elsewhere is how to craft cyber tools in a platform economy to support and sustain skill-based competitive advantage. The most important point is that we are in the midst of a transformation, not the end.

Skeptics like Gordon might ask where is the concrete evidence that this round of innovation will reignite rapid productivity growth similar to the period that ended in the 1970s? There is an array of alternate explanations for the productivity slowdown that is unrelated to technology per se. Our purpose here is not to review or evaluate the rich literature on productivity but suggest the debates that will result from the economic character of the digital transformation.

Central to this discussion, productivity is not simply a technical matter but a real-life story of the reorganization of communities and work to generate new productivity gains. Deploy-

ment of technology is as crucial to productivity as technology itself. One line of argument among economists is that the technology-diffusion machine in the advanced countries has broken down. For example, a 2016 Organisation for Economic Co-operation and Development study¹ found the productivity frontier has been pushed outward but the best practices are not being implemented broadly in the economy. It found the leading 10% of global firms in each sector has had significant and steady productivity increases in the 21st century, while the other 90% continues to lag.¹ The problem for society becomes one of deployment and diffusion, business practices, and structural policy, not the inherent possibilities of the technology.

The OECD results suggesting a gap between the frontier and the rest is still being debated but have raised important questions about the role of intelligent tools in addressing the productivity slowdown. Does the slow productivity growth in the economy as a whole exist because of slow diffusion of leading technology and organizational/business principles? If the diffusion machine is indeed broken, is the reason resistance, overregulation, policy, or incapacity at the level of the individual firm? As Perez²¹ and the Schumpeterians suggest, it might be that productivity moves in jumps, as new paradigms of organization and innovative technologies combine to permit new plateaus, a conclusion that would counsel patience. Each jump to a new plateau implies both production reorganization and new forms of work and work organization.

Are the political obstacles to the diffusion of ICT technology and organizational principles different in this era of intelligent tools from that of steam or electricity? As the Luddites showed in their reaction to the self-acting “spinning mule,” technology deployment and diffusion is rarely a simple or conflict-free process. The mechanization of U.S. agriculture proceeded relatively more smoothly, because the mass production-driven economic growth in the Industrial Midwest and California could absorb the surplus labor. The politics of 21st century growth already involve deep dislocations in

already prosperous well-organized societies that will continue to be difficult politically. Capturing the promise of the technology is as much a political problem as it is a narrowly economic constraint, suggesting policy and political action rather than descent into economic pessimism.

Some economists contend that winner-take-all tendencies in the digital economy are at fault for dislocations.⁵ Are the leading 10% of firms at the productivity frontier because they have dominant market positions unavailable to the other 90%? Along a different line, outsourcing of business services (such as janitorial or even secretarial and bookkeeping) might well keep high-productivity activity in core firms and transfer low-productivity activities to suppliers. If this is the case, the whole system might be no more productive but significantly more unequal.

In sum, we are in the midst of an ICT-powered industrial revolution. The effects emanate from a small set of information-based sectors or leaders at the frontiers of effective deployment. We can decide later whether the period 1970 to 2018 brought as profound a change in our way of life and standard of living as did the period 1870 to 1970. It is clear that the impact of intelligent tools on productivity will depend not just on the technological advances but on the capacity to deploy and diffuse them. It is almost certain that sustainable productivity increases will be a necessary though likely insufficient condition for increasing employment and wages.

Does Work Have a Future?

Consider now the concrete question of jobs and work separately from the abstraction of productivity. Who will work? Who will be employable? What will they do? How might they be compensated? How will labor markets be organized? The jobs question is as difficult to sort through as the productivity question, because it is impossible to predict what new work will arise as the economy changes. Labor markets will be created and transformed by platforms and intelligent tools based on the character and organization of work.

Platforms and labor markets. The current focus on digital platforms and

labor markets has principally considered the ways work is organized and compensated. The emphasis has been on matching work and workers and the belief that increasing numbers of jobs are being converted from stable work to “gig” employment. This logic understates and improperly frames the issue.

Platforms, from Amazon and eBay to Uber and Upwork, and even to YouTube, are built on discovery-and-matching mechanisms, between jobs and employers, clients and contractors, sellers and buyers, and, most abstractly, creators, consumers, and advertisers. The implication is that if only more individuals could participate in the market or if only good matches could be made more easily, growth would accelerate and well being for the vast majority of workers would improve. The premise is that digitization has transformed employment relations between employer and worker (capital and labor). The policy concern here is that moving work to platforms risks facilitating a redefinition of the core of the economy, from employment relations to gig and contract relations.²⁶ Despite contentious debate among scholars and political figures, there is also an argument about how much has really changed over the past few years; for example, one study⁸ suggested that in June 2016 only 0.90% of U.S. adults actively earned income in the “online platform economy.” Are there more such market relationships or just that such relationships are visible now that they are online, rather than signaling a real increase in contingent work?

Academic research on the transfer of work to digital platforms, and the accompanying transformation of once-stable employment to more precarious work or the elimination of entire work categories altogether, while diverse and expanding rapidly, often focuses on a single firm or sector, whether taxis and Uber or encyclopedias and Wikipedia. The current public fixation on Uber and Airbnb is understandable, as they directly challenge two significant traditional industries—transport and lodging. Both involve conversion of consumer goods, cars, and residences into producer goods and thus affect existing labor relationships and markets. If

we extend the scope of consideration to, say, YouTube, which has helped transform the entertainment and self-help-publishing industries or Amazon's self-publishing book business, which is helping reorganize publishing; both convert the labor market relationship to one in which creators "consign" their work to the platform, revealing yet another vector of industry reorganization and with it a labor-process change. Viewed this way, the influence of digital multi-sided platforms on the overall economy is far greater than the narrow focus on Uber and Upwork, and even YouTube, would suggest.

Evaluating the platform economy requires that we project beyond the most evident applications and their effect on the workforce and their employers and consider the ecosystems they organize.

ICT and the reconceptualization of productive activity. Any discussion of work and jobs must consider how production of goods and services will be reorganized as ever more sophisticated ICT is introduced.²⁸ Even as much attention focuses on factories, warehouses are also being automated, and service tasks are being assigned to "smart" programs and robots. One set of arguments, particularly as reported in the popular press, focuses on specific technologies, including AI, robotics, and 3D Printing. A second set, best represented by the now iconic German analysis Industrie 4.0 considers how governments, labor unions, and companies can respond to preserve competitiveness and augment worker skills and capacities, even as the very character of production changes.

A third set of labor-market studies focuses broadly on the consequences of automation and suggests the current digital revolution will indeed generate a world of greater unemployment, more unskilled workers, and greater inequality.⁵ Many of these studies highlight concerns about the destruction and devaluation of work and skills. However, following studies from a number of well-known consulting firms, the conclusions concerning employment are less clear. Implications run from urgency—job tasks for potentially tens of millions of workers will be transformed soon

Komatsu excavator uses computation to calculate the correct angle of its digging blade.



by automation or complacency—as the displacement will be at a scale compatible with ordinary structural change in the economy. The differences in conclusions—from urgency to complacency—depend on the varied judgments of what can be automated and what might be economically feasible to automate, the data sources used to estimate the possible changes, and the timeframe of the structural changes being observed.

The outcomes concerning work and skills ultimately depend on how the new emerging intelligent digital tools are deployed. Moving the technology frontier outward promises new possibilities while eliminating existing ones.⁷ Each set of possibilities often includes distinct implications for value creation and capture. The new frontier, though, does not entirely determine the structures and organizational forms through which a technology might be deployed. If the goal is to, say, reduce carbon emissions, a society can electrify its vehicle fleet, then “decarbonize” the resulting increase in electricity demand with renewable energy, thereby moving to an entirely new energy system. Alternatively, and more in keeping with what the history of technology transitions suggests,² a firm or even a whole society can introduce a transition technology, as the Japanese au-

tomakers Toyota and Honda did with the Prius and Insight hybrids, respectively. Hybrids offer opportunities for improving technologies (such as through batteries and electric-engine systems for automobiles) while staying within the existing carbon-energy system infrastructure and preparing for the expected transition.

Rather than centralized factories or decentralized customization, new approaches are certain to emerge to production organization, and with it new strategies for entrepreneurship and requirements for worker skills. It is possible no single production system will dominate in the 21st century but rather a variety of ways to organize productive activities, as work is continually reconstituted and value chains reconfigured. Mirroring what might become a range of organization models, a remarkable variety of employment arrangements could emerge, too.

Pondering such arrangements leads us back to the question of the effect of intelligent tools on the tasks and work people do for a living. A focus by economists and business leaders solely on the jobs that may be displaced or transformed by intelligent tools hides the opportunities that are certain to emerge and the innovative possibilities that may be unleashed. Whether it is product designers for 3D printers in the maker movement or

video creators on YouTube, new work, tasks, and sources of income are being created. Moreover, the innovation dynamic can never be totally “automated,” remaining for the foreseeable future a domain of human inventiveness and initiative. This is particularly true given that digital resources (such as open source software and cloud computing resources and capital for innovative activities) are more available than ever before.

A crucial question for society, however, is whether this new world will include only employment and reward for the highly trained top 10% of society, those lucky enough to be anointed YouTube “stars,” have their app go viral, start a new firm later acquired by an existing firm, or be employed in a core firm. Where income will come from for those with more modest training and education not blessed with inherited status, born with innate and recognized intelligence, or just not lucky? Some, including Carl Benedikt Frey and Michael A. Osborne²² have argued that broad swaths of work—standard routine tasks, arguably the bulk of work today—are directly vulnerable to displacement by intelligent tools.⁹ However, such displacement is not, in fact, evident. Other research suggests that even routine manufacturing tasks, seemingly most vulnerable to automation, are less routine than they might appear at first glance. Moreover, the automation itself opens new shop-floor-level domains requiring judgment and augmented human capabilities.

An alternate view maintains that computation can augment human intelligence and capabilities. There is already evidence that even routine work can be augmented. Often, however, such augmentation involves contradictory elements. For example, in Japan, where there are shortages today of skilled operators of heavy equipment, equipment manufacturer Komatsu introduced an excavator that uses computation to calculate the correct angle of the digging blade so it does not dig too far. This control enables even relatively inexperienced operators with lower skill levels to work effectively in situations where previously only highly experienced operators could be used.³



The jobs question is as difficult to sort through as the productivity question, because it is impossible to predict what new work will arise as the economy changes.



In any discussion on augmenting human capabilities, the user interface is critical. Programs, websites, and apps are essentially user interfaces and thus augment and empower while structuring human capabilities and activities. Standard office applications (such as Word and Excel) designed and built for personal computers, contributed to the diminishing demand for secretaries and concurrent increase in staff assistants and computer specialists. The user interface profoundly influences who can use and deploy computing power.

Whether and how computer systems augmenting workers’ skills and knowledge will be developed and deployed remains an open question, to be discovered sector by sector, production phase by production phase. Indeed, the required mix of skills will depend on how ICT tools are deployed and on the user interfaces that are developed.

In the choices businesses must make about the design, development, and deployment of the tools they use for automation, one question is crucial: Are workers an asset to be promoted and developed, partners in competition with other firms? If workers are strategic, then a primary challenge is imagining and investing in tools, including user interfaces, that make all workers more productive, effectively a strategy for augmenting intelligence. To illustrate, Ton²⁵ showed that even in the commodity retail business, a profitable strategy can be a good-jobs strategy involving investment in workers and organizational strategies to help those workers develop their capabilities and achieve their potential.

The implication is that if society invests in technologies, business models, and companies subscribing to the belief that intelligent tools will inevitably displace work, with investment after investment made to find ways to substitute capital for labor, then a dystopian outcome is inevitable and with it a road toward digital displacement on a mass scale. The prophecy of ICT displacing human beings will thus be self-fulfilling. In contrast, if a concerted effort is made to discover how to use ICT to augment intelligence, upgrading jobs throughout the work spectrum, then perhaps these digital resources can be harnessed to build a

broadly better future. So government and employers alike must ask: Is there a strategy for using computation to augment human intelligence? And how can we redesign work to leverage human cognition and creativity?

The outcomes depend on societal choices and vision and how technology is deployed and used. Outcomes are not inherent in the technology itself. The balance is yet to be determined. A difficulty is that it is likely easier to identify the specific problems for which intelligent tools can displace jobs than try to understand the ways worker capacity might be augmented. It should be possible to design research initiatives to develop and elaborate a future in which the effect on workers is a key factor to be considered. The continuing progress of intelligent tools will, if it simply displaces work and absent the retraining and creation of new employment opportunities, create significant social upheaval.

To understand the effect of ICT on work tasks and jobs requires that we examine the reorganization of production and the transformation of work itself, as well as labor-market dynamics. It is, in the end, a single woven fabric. If intelligence augmentation requires new skills or integration of work in new ways, who in the platform economy will invest in developing worker skills and encouraging work redesign?

Policy and Politics for the Platform Economy

The sweeping changes brought about by digital technologies are prompting debate throughout society about the institutions and rules of the economy and society.¹⁸ Most fundamental, how will the benefits of the promised new productivity be shared among all members of society? The political question is: What sort of world is emerging, as platforms and intelligent tools continue to progress?

The policy agenda is long and diverse, so consider the following comments to help organize the discussion. In the event of technological shifts as large as this one, various sectors and regulatory issues are affected, but the ongoing debate and discussion are siloed, despite the fact that decisions in one regulatory realm inform and in-



The innovation dynamic can never be totally “automated” and remains for the foreseeable future a domain of human inventiveness and initiative.



fluence developments and technological trajectories in others.

We note two policy categories:

Platform governance. The increasing power of the firms that own platforms raises the question of how to define the tension between private power and public governance. Far more than with most previous industries, digital platforms are regulatory structures. Even more than in natural monopolies (such as electric and water utilities), today's digital platforms deeply structure the rules and parameters of action available to users. The classic insight in this regard was by American lawyer and constitutional scholar Lawrence Lessig who titled the first chapter in his 2006 book *Code*¹⁷ “Code Is Law”; that is, governance is effectively embedded in the code itself. Firms can introduce platforms that directly or indirectly circumvent existing regulations. If the new service is adopted, as was the case with both Uber and Airbnb, the result can be a direct challenge to state regulatory authority. When the platform occupies an unregulated market or a market in which existing regulations are unclear and difficult to apply, then new platform businesses often compel consideration of new regulations, or, at minimum, new regulatory interpretations. For example, should Airbnb landlords be subject to the land-use regulations and disability-access regulations that apply to hotels? Moreover, platform-based private rule-making in the form of code creates rules that are generally hidden and not available to users or governments for discussion or alteration. These platforms have remarkably powerful social effects. More generally, the choice, and implicitly the debate, is whether platforms and platform businesses should be treated as abstract technologies, technology businesses, or ordinary participants in the particular sectors, whether transportation—Uber—or accommodations—Airbnb? In contrast, Amazon would contend it merely provides logistics support to itself and the users of its platform.

Managing the tension between public interest and private-platform strategies requires that historically siloed and separated debates be integrated into policy discussions. In practice, however, questions about big data, privacy, and security are in-

timately connected. For example, the voice-activated digital helpers from Amazon and Google not only have privacy implications but, because they recommend products and services, also affect marketplace competition.²⁴ Further, their payment systems could also raise banking regulatory questions. Digital helpers are bound to produce further vertical integration that could also require regulatory intervention. Decisions in one regulatory area can directly influence decisions in other regulatory areas.

The greatest strategic advantage for platform firms is their algorithms and the data they collect. Not surprisingly, these firms claim their algorithms and data are trade secrets not be subject to public scrutiny.

Intelligent tools. To establish a technology trajectory in which intelligent tools contribute to human creativity, one priority for business leaders should be to consider how harnessing computer-human complementarities might create advantage in ways that will be valued and help generate success in the marketplace. Society should thus fund research projects aimed at identifying where, how, and why intelligent tools contribute to augmentation of human capabilities. This research should make possible inferring the kinds of applications and deployments best suited to computer-human collaboration and encourage their development and deployment. Identifying alternatives is difficult. Even more difficult is how to develop organizational strategies that support worker development, augment human capabilities, and amplify human intelligence.

Conclusion

Politics translates debate into social and economic policy. Business leaders, political figures, and workers need to resolve the politics and economics of structural change caused by the movement of social life and economic activity onto ICT platforms and the effect on employment and the work process. In some instances, as with Germany's Industrie 4.0, there will be a coherent national debate, while in others (such as policy in response to, say, Amazon's dominance of online retail) such debate may be

difficult to formulate and responses to organize. Policy and politics will be an important force shaping the consequences of the increasing penetration of platforms and other intelligent tools into the fabric of everyone's economic and social life. As existing sectors decline or are transformed, new market leaders will emerge, displacing existing firms, even as new domains and sectors appear. The existing workforce will transform or be pushed aside as new forms of work and new strategies for organizing the production and distribution of goods and services are introduced. There is already a struggle over governance between the public rules and governance embedded in platform algorithms and code. We hope this article provides a framework for a discussion that is only beginning.

Acknowledgments

The authors contributed equally to this article. We gratefully acknowledge the financial support of the Kauffman Foundation and helpful comments of Roger Bohn, Stuart Feldman, Ken Goldberg, Kenji Kushida, Niels Christian Nielsen, Hanne Shapiro, Shankar Sastry, Costas Spanos, and Laura Tyson. We thank the anonymous reviewers for their penetrating comments. All arguments advanced and conclusions herein are solely the responsibility of the authors. C

References

1. Andrews, D., Criscuolo C., and Gal, P. N. *The Best Versus the Rest: The Global Productivity Slowdown, Divergence across Firms and the Role of Public Policy*. Technical Report. Organisation for Economic Cooperation and Development, Dec. 2, 2016; http://www.oecd-ilibrary.org/economics/the-best-versus-the-rest_63629cc9-en;jsessionid=9ag8ukcclm7fb.x-oecd-live-03
2. Arthur, W.B. *The Nature of Technology*. Simon & Schuster, Inc., New York, 2009.
3. Asada, H. *Partnership-Driven Business Growth in Komatsu: Autonomous Trucking and Smart Construction*. PowerPoint Presentation. International Partnerships for Advanced Intelligent Systems at Stanford University, Stanford, CA, Oct. 22, 2015; <http://asia.stanford.edu/us-atmc/wordpress/wp-content/uploads/2015/10/151022-Komatsu-Slides.pdf>
4. Autor, D.H. *Polanyi's Paradox and the Shape of Employment Growth*. Working Paper 20485. National Bureau of Economic Research (Sept. 2014); <http://dx.doi.org/10.3386/w20485>
5. Brynjolfsson, E. and McAfee, A. *The Second Machine Age*. W.W. Norton & Company, New York, 2014.
6. David, P.A. The dynamo and the computer: An historical perspective on the modern productivity paradox. *American Economic Review* 80, 2 (1990), 355–361.
7. Dosi, G. Technological paradigms and technological trajectories: A suggested interpretation of the determinants and directions of technical change. *Research Policy* 11, 3 (1982), 147–162.
8. Farrell, D. and Greig, F. *The Online Platform*

Economy: Has Growth Peaked? J.P. Morgan Chase & Co. Institute, New York, Nov. 2016; <https://www.jpmorganchase.com/corporate/institute/document/jpmc-institute-online-platform-econ-brief.pdf>

9. Frey, C.B. and Osborne, M.A. The future of employment: How susceptible are jobs to computerisation? *Technological Forecasting and Social Change* 114 (2017), 254–280.
10. Gawer, A. and Cusumano, M.A. *Platform Leadership*. Harvard Business School Press, Boston, MA, 2002.
11. Gordon, R.J. *The Rise and Fall of American Growth*. Princeton University Press, Princeton, NJ, 2016.
12. Kagermann, H., Wahlster, W., and Helbig, J. *Securing the Future of the German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative Industrie 4.0. Final Report of the Industrie 4.0 Working Group National Academy of Science and Engineering*. Acatech, Germany, Apr. 2013.
13. Kenney, M. and Zysman, J. The rise of the platform economy. *Issues in Science and Technology* 32, 3 (Spring 2016), 61–69.
14. Khan, L. Amazon's antitrust paradox. *Yale Law Journal* 126, 3 (Jan. 2017), 710–804.
15. Kushida, K.E., Murray, J., and Zysman, J. Cloud computing: From scarcity to abundance. *Journal of Industry, Competition and Trade* 15, 1 (Mar. 2015), 5–19.
16. Lazoniak, W. Profits without prosperity. *Harvard Business Review* 92, 11 (Sept. 2014), 47–55.
17. Lessig, L. *Code: Version 2.0*. Basic Books, New York, 1999.
18. Lobel, O. The law of the platform. *Minnesota Law Review* 101 (2015), 87–166.
19. Nooren, P., van Gorp, N., and van Eijk, N. Digital platforms: A practical framework for evaluating policy options. In *Proceedings of the Internet, Policy & Politics Conference* (Oxford, U.K., Sept. 22–23). The Platform Society, Oxford, U.K., 2016.
20. Parker, G.G., Van Alstyne, M.W., and Choudary, S.P. *Platform Revolution*. W.W. Norton & Company, New York, 2016.
21. Perez, C. *Technological Revolutions and Financial Capital*. Edward Elgar Publishing, Cheltenham, U.K., 2003.
22. Pfeiffer, S. Robots, Industry 4.0 and humans, or why assembly work is more than routine work. *Societies* 6, 2, (May 2016), 16.
23. Schumpeter, J.A. *The Theory of Economic Development*. Transaction Publishers, New York, 1934.
24. Stucke, M.E. and Ezrachi, A. *How Your Digital Helper May Undermine Your Welfare, and Our Democracy*. University of Tennessee Legal Studies Research Paper No. 324; https://papers.ssrn.com/Sol3/papers.cfm?abstract_id=2957960
25. Ton, Z. *The Good Jobs Strategy*. Houghton Mifflin Harcourt Publishing, New York, 2014.
26. Weil, D. *The Fissured Workplace*. Harvard University Press, Cambridge, MA, 2014.
27. Zysman, J. The algorithmic revolution: The fourth service transformation. *Commun. ACM* 49, 7 (July 2006), 48.
28. Zysman, J., Feldman, S., Kushida, K.E., Murray, J., and Nielsen, N.C. Services with everything: The ICT-enabled digital transformation of services. Chapter in *The Third Globalization? (First Edition)*. Oxford University Press, Oxford, U.K., 2013, 1–23.

John Zysman (Zysman.john@gmail.com) is a Professor Emeritus in the Department of Political Science, University of California, Berkeley, Berkeley, CA, cofounder of the Berkeley Roundtable on the International Economy, and convener of the Berkeley Project Work in an Era of Intelligent Tools and Systems.

Martin Kenney (mfkenney@ucdavis.edu) is Distinguished Professor of Human and Community Development at the University of California, Davis, and Senior Project Director at the Berkeley Roundtable on the International Economy; he is also an Affiliated Faculty at Instituto di Management at the Scuola Superiore Sant'Anna, Pisa, Italy.

©2018 ACM 0001-0782/18/2



Watch the authors discuss their work in this exclusive *Communications* video. <https://cacm.acm.org/videos/the-next-phase-in-the-digital-revolution>

Beta testers should represent a future product's target users as much as possible.

BY VLASTA STAVOVA, LENKA DEDKOVA,
MARTIN UKROP, AND VASHEK MATYAS

A Large-Scale Comparative Study of Beta Testers and Regular Users

BETA TESTING IS an important phase of product development through which a sample of target users (potential adopters) try a product ahead of its official release. Such testing is practically ubiquitous; in every kind of company, from medicine to software development, participants test and troubleshoot products to help improve their performance and avoid defects.

Companies should care who their beta testers are. In order to generalize beta-testing outcomes, the population of testers must be as representative of the target users as possible. If not, results of the testing could be biased and fail to capture important product flaws; that is, beta testing for different purposes demands different

sets of beta testers. Such aspects of beta testing are often underestimated by software developers; their companies often use any beta tester available, without proper selection, and later without analyzing to what extent the testers were comparable to the population of (targeted) users. Though it was once easier for companies to know their beta testers well,⁶ this is not the case today due to the vast reach of the Internet and quickening pace of releasing updates and new versions. They should indeed pay more attention to their testers, selecting wisely, because appropriate beta testing is more efficient and economical than later potential failure of a new product.

The costs of poor beta testing were apparent from the beginning of software development. An early example, from the 1990s, is a software company that chose only one site for its testing.⁶ Based on the results, developers then made several changes to the product being developed. Since the beta testers represented only a specific subpopulation of intended users, the product became so customized it could not be marketed to other organizations. For example, in 2012, the Goko company released a web portal for developing multiplayer games but used a pool of beta testers so small it did not notice a serious bug connected with site load as the portal became popular.²² Moreover, beta testing is not only about bug hunting; benefits also include enhanced product support and marketing.⁶

Here, we present case-study results of a comparison of beta testers and regular users for an online security prod-

» key insights

- The fewer testers a company has, the pickier it should be about their selection.
- Testers should be representative of the company's regular users, and the company should keep checking that this is the case or pursue further analysis.
- For products designed for international customers, the company needs to focus on country differences among testers and regular users to avoid potential localization conflicts.



IMAGE BY G-STOCK STUDIO

uct we conducted in 2015 and 2016. We analyzed the records of nearly 600,000 participants worldwide, aiming to determine whether the beta testers represented regular users well enough. Despite the fact that alpha testers are well described in the software-development literature, as far as we know, no larger field study comparing beta testers and regular users had ever been published. We thus present what we believe is the first public large-scale comparison of beta testers and regular users.

We investigated whether companies should be more selective about

their beta testers or simply take an intuitive approach that says, “The more testers, the better the result.” We did not aim to investigate goals or parameters and conditions of beta testing and began with three main research questions:

Do the subsamples have similar profiles with respect to hardware and operating system?;

Do the subsamples reflect similar age, gender, and education profiles? What about cultural background?; and

Do the subsamples see themselves as equally skilled regarding their use

of computers and perceive their data as safe?

Here, we review published research in beta testing, describing methods and analyzing data, then map the results for the three questions, and finally discuss study limitations and contributions and actionable takeaways (see the sidebar “Actionable Takeaways” on page 44).

Study Rationale

Testing represents 30% to 50% of the cost of software development¹ and approximately 50% of development

time.¹⁷ There are many testing phases, with the first usually involving alpha testers. The number of potential testers is limited by a company's size, and even for big companies, it is impossible to duplicate all possible hardware/software configurations. Whereas alpha testers are typically company employees, beta testers are the first product users outside the company, and their feedback can greatly influence product

design before the product can be used by paying customers.

Tapping the universal scope of the Internet, thousands of beta testers with different devices and practices can report on a product before its official release. An additional benefit follows from being able to include test subjects in multiple countries. Since beta participants can come from many different locations, potential localization issues (such as language, currency,

culture, and local standards) can be identified and included.²² Moreover, cultural context also affects a new product's perceived usability.²⁴ Beta testers thus bring huge benefits to the development process by detecting potential hardware conflicts and performing usability checking.

While many alpha- and beta-testing studies have been published, the idea of comparing beta testers and regular users had only rarely been tackled when we began. For example, Mantyla et al.¹³ investigated the related question "Who tested my software?" but limited themselves to the employees of only three companies. Other studies^{11,14} yielded insight into the software-tester population yet were based mainly on specific subpopulations (such as people interested in testing, users of specialized forums, and LinkedIn participants) or company employees, so a selection bias could have occurred.

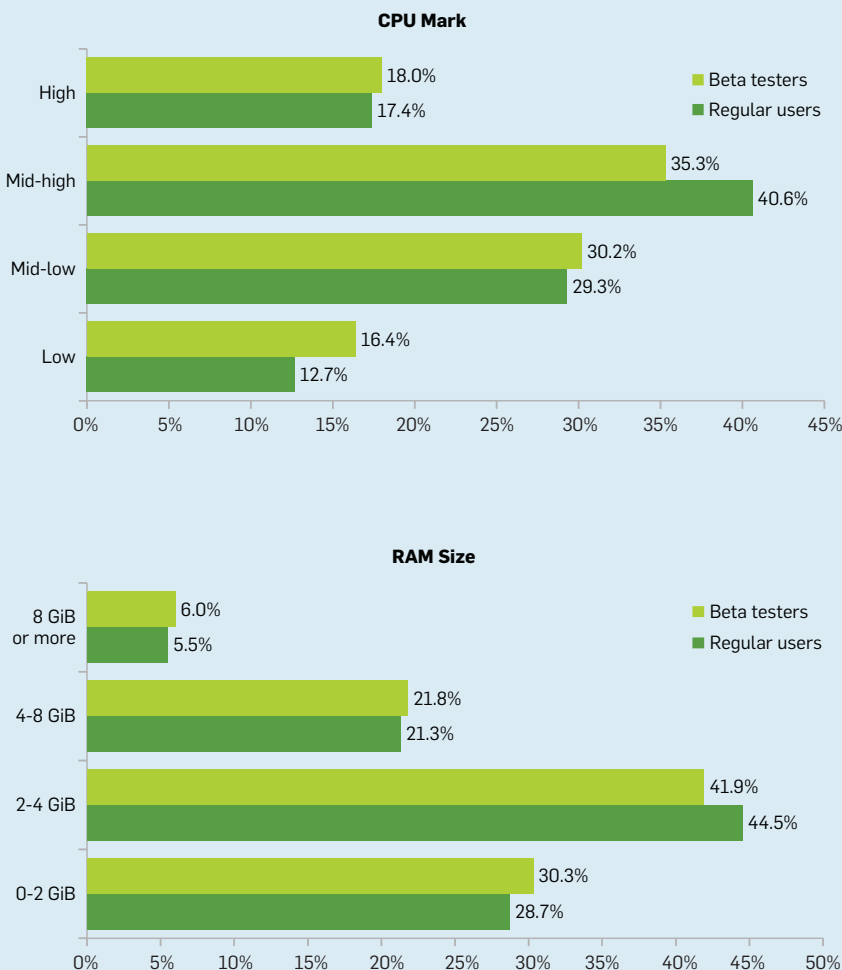
We compared beta testers and regular users in various aspects of software use and testing, starting with technology. Having similar devices with regard to, say, hardware and operating systems is a basic requirement for successful software beta testing. Since physical environment influences usability testing,²⁰ the device used to test an application could also influence its usability. For example, security software running in background can decrease the perceived overall performance of the machine it is running on and thus its perceived usability. Participants with low-end hardware could encounter different usability issues compared to those with high-end hardware. Beta testers are viewed as problem solvers or early adopters¹⁹ with access to the most up-to-date computer hardware.

We also examined user demographics. Earlier research had reported that regular users' IT-related behavior is affected by gender, age, education, and cultural background. For example, Dunahay et al.⁸ found that a greater rate of computer use and online activity was associated with lower age, higher education, and being male.⁸ The differences in IT usage are also related to country of origin.¹⁶ Countries differ in terms of the state of their national information society culture, leading to varying access opportunities and creating digital disparities among nations.^{2,5} As a

Table 1. Overview of participant numbers (following data cleaning).

	Unique devices	Completed questionnaires	
Beta testers	77,028	5,514	7.2%
Regular users	499,142	24,084	4.8%
Total	576,170	29,598	5.1%

Figure 1. Basic hardware characteristics for beta testers and regular users; see also the section on study limitations.



result, the populations of some nations could be more computer savvy and/or inclined to use free software, even while still in beta. For example, anecdotal evidence suggests Japanese users take up emerging technologies more slowly than users in other countries.¹⁵

Varying patterns of Internet/computer use are also associated with users' computer self-efficacy and attitudes concerning privacy. Computer self-efficacy⁴ reflects the extent to which users believe they are capable of working efficiently with a computer. Users with a greater confidence in their computer skills tend to use computers more,⁴ adopt new technology quicker,^{10,23} and achieve better performance in computer-related tasks.⁷ Regarding privacy perception, marketing research consistently shows how consumers' online behavior (such as willingness to provide personal information or intention to use online services) is affected by concern over privacy.^{12,21} Since beta testing usually includes sharing one's system settings, location, or even personal information with the testing company, it may discourage users with more strongly held privacy concerns or those who store more private data on their computers. However, discouraged potential testers could still be an important segment of the end-user population, with distinct expectations for the final product.

Methodology

We conducted our study with ESET (<https://www.eset.com>), an online security software company with more than 100 million users in more than 200 countries and territories,^a using two samples for analyses: beta testers and regular users of a line of ESET security-software solutions for Windows.

The ESET beta program allowed anyone to download the beta version of a product and become a public beta tester. Despite the fact that users had to complete and return a questionnaire before they could beta test the product, ESET uses no special criteria when selecting its beta testers. ESET beta testers report bugs and/or suggest improvements, motivated by the opportunity to use a beta product for free, possibly sooner than regular users.

a <https://www.eset.com/int/about/>



We analyzed the records of nearly 600,000 participants worldwide, aiming to determine whether the beta testers represented regular users well enough.



We collected our sample of beta testers ($N = 87,896$) from June 2015 to December 2015 and the sample of regular users ($N = 536,275$) from January 2016 to March 2016. We first collected anonymized system parameters for each ESET installation, including processor configuration, RAM size, operating system, country, and time spent on each installation screen. We identified countries through the GeoIP2^b database (<https://www.maxmind.com/en/geoip2-databases>). A single data record represented a single installation of the software.

We gave a questionnaire to users at the end of the installation process, saying that filling it out was voluntary; we offered no incentives other than to say that completing it will help ESET improve its products. A total of 6,008 beta testers completed at least one questionnaire item (7.80%), along with 27,751 regular users (5.56%). The questionnaire was in English, and we collected no identification data. The questionnaire was also a source for collecting demographic data and privacy perceptions.

Data cleaning. We cleaned the data to remove tester and user information associated with ESET's internal IP space domain (0.282% of the sample), ensuring we would exclude ESET's own alpha testers. Moreover, since each data entry reflected only a single installation, duplicate entries could potentially have come from the same device. To inhibit bias, we identified cases with the same combination of hardware specification and IP address, randomly selected one, and deleted the rest, thus removing 7.429% of beta tester and regular user data.

We presented the whole questionnaire on four screens, using the time testers and users spent on each screen to clean the data; we considered as invalid testers and users who spent less than six seconds on a screen with two items and those who spent less than seven seconds on a screen with three items, omitting their data from our analyses; $N = 10,151$, or 30.1%, of questionnaire respondents.

The final cleaned sample for the study thus included 576,170 installations on unique devices, including 29,598 questionnaires with at least one answered item (see Table 1).

b <https://www.maxmind.com/en/geoip2-databases>

Figure 2. Beta testers and regular users compared with respect to operating system versions.

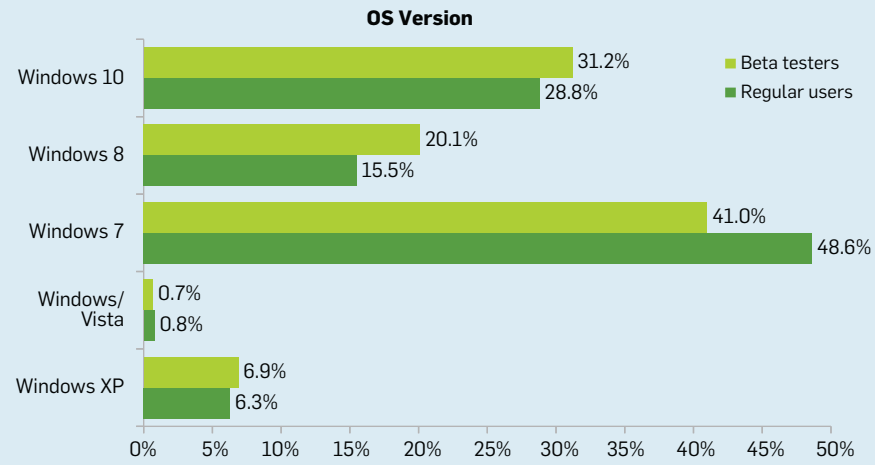


Figure 3. Beta testers and regular users compared with respect to the continent where they live.

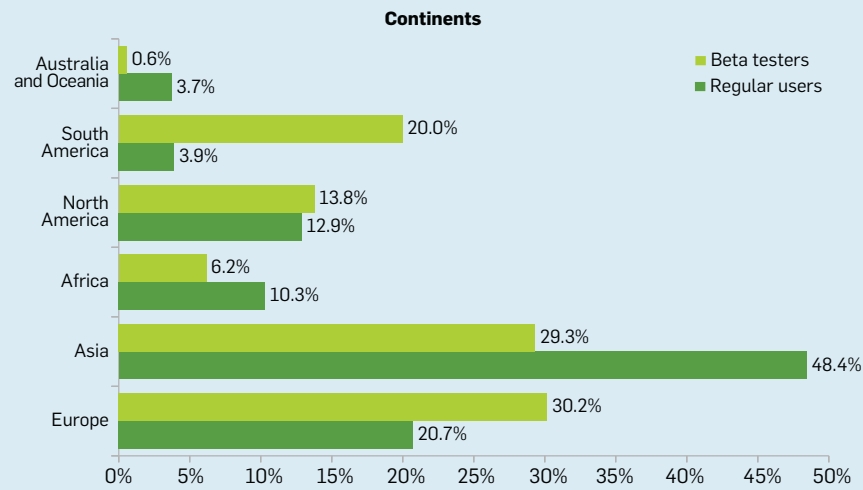


Table 2. The most represented countries in the subsamples of beta testers and of regular users.

Regular users			Beta testers		
Country	N	%	Country	N	%
Iran	81,035	16.2	Mexico	5,662	7.4
U.S.	50,220	10.1	Indonesia	5,117	6.6
India	26,532	5.3	Brazil	4,251	5.5
Indonesia	25,959	5.2	China	4,132	5.4
U.K.	25,173	5.0	Peru	3,422	4.4
Egypt	21,649	4.3	Russia	3,348	4.3
Romania	16,582	3.3	Ukraine	2,979	3.9
Pakistan	15,831	3.2	Spain	2,513	3.3
Peru	15,280	3.1	Egypt	2,393	3.1
Philippines	14,904	3.0	<unknown>	2,314	3.0
South Africa	13,951	2.8	Iran	2,306	3.0
UAE	11,584	2.3	India	1,771	2.3
Thailand	10,719	2.1	Argentina	1,679	2.2
Australia	10,621	2.1	U.S.	1,560	2.0
Germany	8,259	1.7	Poland	1,543	2.0

Analytical strategy. We used the χ^2 test (categorical data) and t -tests (interval data) to assess the differences between beta testers and regular users; analyses on large samples typically show statistically significant results even for very small effects. When considering such results, it is important to interpret effect size rather than significance alone. We thus calculated Cramer's V (φ_c) for categorical data and Cohen's d for interval data. For φ_c , the value of 0.1 is considered small, 0.3 medium, and 0.5 a large effect size, and for d , the respective values are 0.2, 0.5, and 0.8.^{3,9}

The fact that our questionnaire data came from only a subsample of users could suggest possible bias in our results (see the section on study limitations). For insight into the differences between the samples with and without the questionnaire, we compared users with regard to the parameters available for them all, including platform information, CPU performance, RAM, and OS version. We found the effect of the differences to be negligible ($\varphi_c < 0.034$). We are thus confident the questionnaire data was valid and informative, despite having been obtained from only a small subsample of users.

Technology

We first looked at the technology, including hardware platform (32 bit or 64 bit), CPU model, RAM size, and OS version.

Hardware. The platforms running ESET software differed only slightly between subsamples; 35.3% of beta testers used 32-bit systems, while approximately 34.5% of regular users used 32-bit systems; $\chi^2(1) = 20.998$, $\varphi_c = -0.006$, $p < 0.001$, and $N = 576,170$.

We categorized CPU performance into four groups—low-end, mid-low, mid-high, and high-end—based on the PassMark CPU Mark criterion.¹⁸ We matched CPU name against the PassMark online database. Since CPU names are not standardized, we were unable to assign the score in 3.040% of the cases; $N_{noCpuMark} = 17,514$, distributed proportionally among beta testers and regular users.

The beta testers were more represented in the low-performance category and regular users in the mid-high category. The proportions were notably

similar in the mid-low and high-end categories (see Figure 1). Although statistically significant, the effect was small; $\chi^2(3) = 1187.546$, $\varphi_c = 0.045$, $p < 0.001$, and $N = 576,170$.

We likewise grouped RAM size into four categories: 0–2GB, 2–4GB, 4–8GB, 8GB, and >8GB. Regular users' proportion was greater in the 2GB–4GB category, while beta testers dominated in the lowest, or 0GB–2GB, category. The proportions in the two largest-size categories were similar, as in Figure 1. The small size of the effect suggested differences were negligible, despite being significant $\chi^2(3) = 206.926$, $\varphi_c = 0.019$, $p < 0.001$, and $N = 576,170$.

Operating system. Beta testers predominated in the two most current OS versions at the time—Windows 8 and Windows 10—while regular users predominated in Windows 7, with nearly equal representation using Windows Vista and XP (see Figure 2). The size of the effect was again small at $\chi^2(2) = 1,925.745$, $\varphi_c = 0.058$, $p < 0.001$, and $N = 575,979$. Other Windows versions (such as Windows 98 and Windows 2000) were also marginally represented but omitted due to the extremely low counts; that is, $<0.001\%$, $N_{\text{otherWinVersions}} = 191$. Note the study targeted only users of Microsoft Windows software.

We found that Windows 10 was more often used by beta testers than by regular users, even though we collected their data sooner; regular users in the survey thus had more time to upgrade, indicating beta testers are often recruited from among early adopters.¹⁹

Specific configurations. We were also interested in specific configurations of users' devices. We combined all four technological aspects, including OS platform, CPU performance, RAM size, and OS, to help us identify 116 unique hardware+software combinations in the dataset of 43,519, or 7.556% of the total sample. The sample of regular users included 114 combinations; we found two specific combinations among beta testers' devices not find among regular users, and the sample of beta testers included 102 combinations. However, the combinations not present among beta testers were only marginally present among regular users at $N_{\text{onlyStandard}} = 52$, or 0.010%, leading us to conclude that for almost every regular user in the

sample, there was a beta tester in the sample with the same combination of examined parameters.

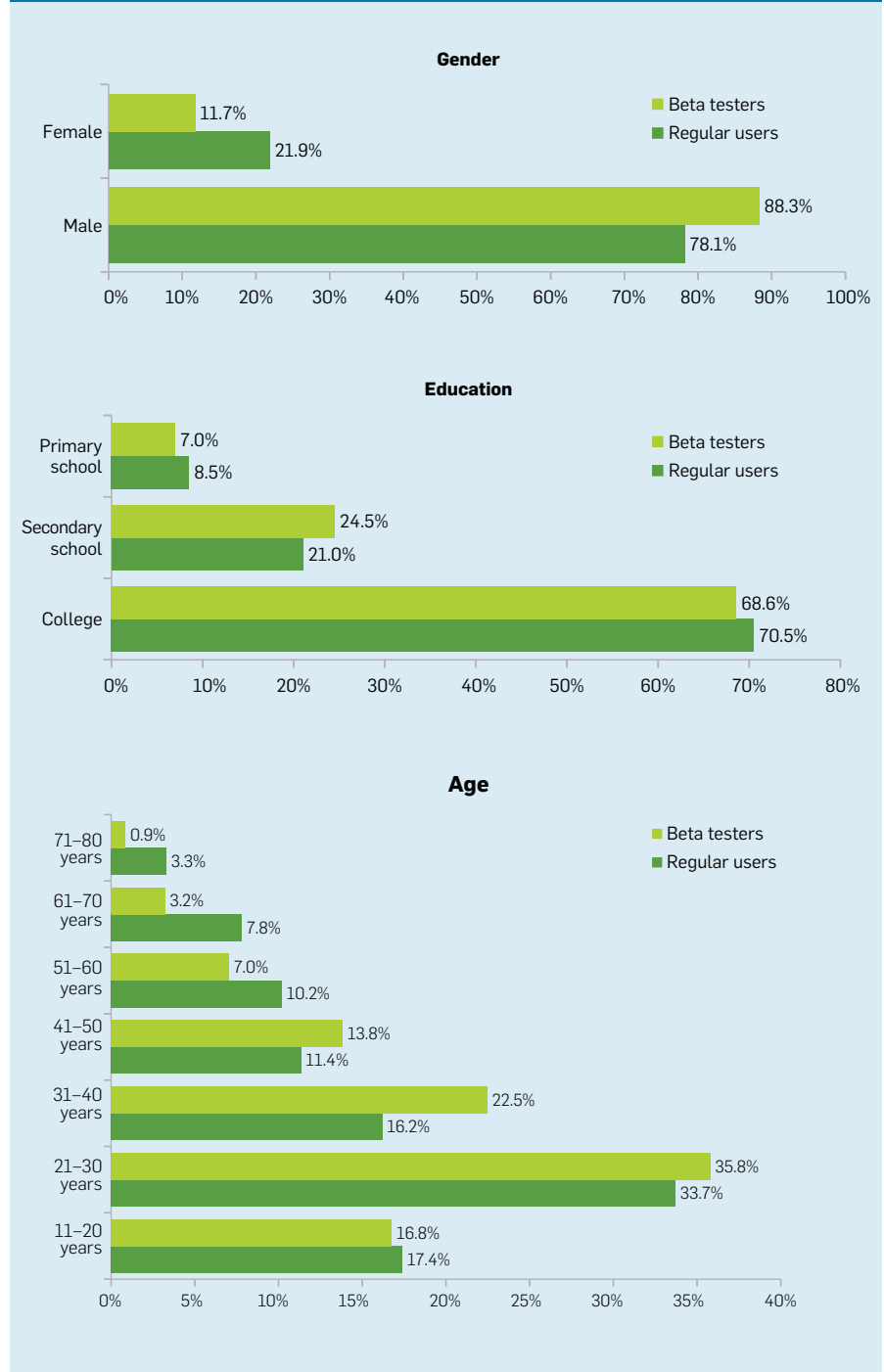
Demography

Here, we discuss participants' cultural and demographic profiles, focusing on country of origin, gender, age, and attained education.

Country of origin. As noted, we based a participant's country on the GeoIP2 database search, a procedure

that failed to assign a country to 0.4% of participants $N_{\text{noCountry}} = 2,408$. We grouped countries by continents and compared the two subsamples (see Figure 3), observing significant differences, notably that beta testers substantially predominated in South America and Europe, while regular users were more often based in Asia, Africa, and Australia/Oceania, with $\chi^2(5) = 39,049.72$, $\varphi_c = 0.261$, $p < 0.001$, and $N = 573,538$; see Table 2 for detailed information regarding the study's most

Figure 4. Demographics of the subsamples compared based on questionnaire data.



represented countries. Only Iran, India, Egypt, and the U.S. were represented in both subsamples.

ESET has subsequently begun to investigate these issues with respect to product localization and usability, where country differences likely play a role.

Gender and age. Figure 4 includes basic information regarding demography. In both subsamples, males represented the vast majority of study participants, though there were more females among regular users than among beta testers, with $\chi^2(1) = 277.493$, $\varphi_c = 0.099$, $p < 0.001$, and $N = 28,328$.

Regular users were on average older than beta testers, with $M_{\text{beta}} = 32.96$, $SD = 12.974$; $M_{\text{standard}} = 35.74$, $SD = 16.327$; $t\text{-test}(25,938) = 11.108$; $p < 0.001$; $d = 0.195$, and $N = 28,940$. Due to the wide range of ages among study participants—11 to 80—we categorized all ages into seven groups to analyze the differences in more informative ways, as in Figure 4. For example, there were significantly more beta testers than regular users ages 21 and 50, while the opposite applied to other categories, with $\chi^2(6) = 366.286$, $\varphi_c = 0.119$, $p < 0.001$.

Education. Education attainment reflected a consistent pattern in both subsamples, with college being represented most and primary school least. The pattern was consistent even when we omitted the youngest users, or those who could not have yet reached higher education. Beta testers were more represented in secondary education than regular users, but the effect size was small, with $\chi^2(6) = 237.085$, $\varphi_c = 0.038$, $p < 0.001$, and $N = 26,354$.

Other demographic insights. We combined the demographic data of study participants to determine how well beta testers also represented various demographic segments of regular users. Combining seven categories of age, gender, and education helped us identify 42 unique combinations. Only two were present in the sample of regular users (none among beta testers), both female, ages 71 to 80, one with primary ($N_{\text{standard}} = 4$), the other with college education ($N_{\text{standard}} = 109$). Remaining combinations were present in both subsamples, with a fairly similar distribution. The greatest difference we found was among males, ages 31 to 40, with college education, who were represented more often

among beta testers (14.172%) than among regular users (9.539%).

Computer Self-Efficacy and Privacy Perception

We assessed users' computer self-efficacy and privacy perceptions through dedicated questions in an optional questionnaire, covering installation-related actions like displaying the target installation folder.

Each ESET software installation included an option for changing installation folder. Beta testers and regular users thus had to click on the "change installation folder" link on one of the screens during the installation process to go to the respective screen. This action was also the only way a user could see the default installation folder, not otherwise displayed. Only a few participants did this, with beta testers visiting the screen more than twice as often as regular users, with 1.1% of regular users and 2.6% of beta testers. This difference was statistically significant, though the effect size was negligible, with $\chi^2(1) = 1215.180$, $\varphi_c = 0.046$, $p < 0.001$, and $N = 576,170$.

Computer self-efficacy and digital skills. We included two questions to help us assess users' digital skills:

Do you consider yourself a skilled computer user? Likert scale from 1 (not at all skilled) to 6 (extremely skilled); and

Regarding this computer, are you an IT technician? Y/N. Participating beta testers were more often IT technicians, with $\chi^2(1) = 285.988$, $\varphi_c = 0.110$, $p < 0.001$, and $N = 23,607$, judging themselves more skilled than regular users, at $M_{\text{beta}} = 4.46$, $SD = 1.313$; $M_{\text{standard}} = 4.18$; $SD = 1.473$; $t\text{-test}(22,631) = -11.743$; $p < 0.001$; $d = 0.200$; and $N = 22,633$.

Privacy perceptions. The last part of the questionnaire asked about how private data is stored in users' computers, how sensitive users are regarding their privacy, and users' beliefs about the computer being generally a safe device. We measured all items on a six-point Likert scale ranging from 1 (not at all) to 6 (extremely private/sensitive/safe) by asking:

- Do you consider the data in this computer private?;
- In general, are you sensitive about your privacy?; and
- In general, do you consider computers to be safe devices against

online attacks (such as viruses, hacking, and phishing)?

Beta testers and regular users alike reported the same average level of private data in their computers, with $M_{\text{beta}} = 4.678$, $SD = 1.419$; $M_{\text{standard}} = 4.690$, $SD = 1.560$; $t\text{-test}(24,323) = 0.504$; $p = 0.614$, and $N = 24,325$, and both quite similar in being privacy sensitive, with $M_{\text{beta}} = 4.755$, $SD = 1.376$; $M_{\text{standard}} = 4.809$; $SD = 1.492$; $t\text{-test}(23,976) = 2.272$; $p < 0.05$; $d = 0.037$, $N = 23,978$. We found only one small difference in their evaluations of general computer safety: Beta testers considered computers slightly safer than did regular users, with ($M_{\text{beta}} = 4.098$, $SD = 1.712$; $M_{\text{standard}} = 3.902$; $SD = 1.819$; $t\text{-test}(23,832) = -6.784$; $p < 0.001$; $d = 0.111$, $N = 23,834$). We observed that beta testers consider themselves more skilled as IT users and the computer as a safer device than do regular users. This might suggest they were aware of security risks associated with computer use and felt capable of addressing them.

Study Limitations

Some limitations beyond our control could have influenced these results. Despite our careful cleaning process, we could not be completely sure that each record corresponded to a unique participant/device. For example, the OS version was based on the Windows system variable "current version" that did not differentiate end user and server products. However, we assumed the number of servers in the study was negligible, as the installed base of ESET systems was, at the time, designed for end-user devices. We also lacked details of participants' devices, technological measures that might have shown more nuanced configuration discrepancies.

The relatively small ratio of users completing the questionnaire could also have represented other limitations. First, self-selection and non-response bias might have skewed our results. For example, most study participants reported at least some college education and could have thus been expected to be able to recognize the value of user feedback better and be more willing to complete a product-related questionnaire. However, they did not differ in terms of hardware or software from those skipping the questionnaire altogether. We had only a few

Actionable Takeaways

Our research produced the following actionable takeaways for software developers:

Using data. Data you can collect can help you learn who your users and beta testers are; consider country of origin, software and hardware configuration, and basic demographics;

Selecting testers. The fewer testers you have, the pickier you should be about their selection;

Identifying usability issues. When testing international products, ensure beta testers are culturally representative of regular users to help identify potential localization and cultural usability issues; and

Ensuring representation. Most important, testers should be representative of regular users; keep checking that this is the case or pursue additional rigorous analyses to reach the most credible and applicable conclusions possible.

options for validating participants' answers. Despite the thorough cleaning, some flawed questionnaire answers could have remained. Also, writing the questionnaire in English could have discouraged users not proficient in that language.

The datasets of participating beta testers and regular users included different numbers of participants and were collected at different times. This could have influenced the number of participants using, say, Windows 10, as the study was conducted during a free-upgrade period. Moreover, the research was based on only the English versions of the software, missing customers who prefer other languages.

Conclusion

Working with security-software firm ESET, we conducted a large-scale comparison between beta testers and regular users of ESET's main product. We focused on technological aspects of ESET's user demographics and nearly 600,000 users' self-reported computer self-efficacy.

The participating beta testers were early adopters of newer operating systems, and their distribution was significantly skewed toward the most current versions at the time, despite

having limited time for Windows 10 migration. They also tended to be younger, more often male, and perceived themselves as more skilled with their computers and also more often IT technicians, supporting the "beta testers as geeks" stereotype. However, their hardware—platform, CPU performance, and RAM size—was similar to that of regular users, somewhat contradicting the popular image.

We found a striking difference in their countries of origin; from the top 10 most represented, only three appeared in both subsamples.

Overall, study beta testers represented regular users reasonably well, and we did not observe a regular-user segment that would be underrepresented among beta testers. ESET's approach of not filtering beta testers and "the more testers the better" followed by analyses of selected observed differences seems sufficient for developing its software products. For large international companies able to attract large numbers of beta testers, this may be the most efficient approach. However, for smaller, local, or less-well-established companies, this approach would probably not yield representative outcomes and could even shift development focus in a wrong direction.⁶

For more, including a video, see <http://crcs.cz/papers/cacm2018>

Acknowledgments

We thank Masaryk University (project MUNI/M/1052/2013) and Miroslav Bartosek for support and to the anonymous reviewers and Vit Bukac for valuable feedback. 

References

1. Biffi, S., Aurum, A., Boehm, B., Erdogmus, H., and Grünbacher, P. *Value-Based Software Engineering*. Springer Science & Business Media, Berlin, Germany, 2006.
2. Chinn, M.D. and Fairlie, R.W. ICT use in the developing world: An analysis of differences in computer and Internet penetration. *Review of International Economics* 18, 1 (2010), 153–167.
3. Cohen, J. *Statistical Power and Analysis for the Behavioral Sciences, Second Edition*. Lawrence Erlbaum Associates, Inc., 1988.
4. Compeau, D.R. and Higgins, C.A. Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly* 19, 2 (June 1995), 189–211.
5. Cuervo, M.R.V. and Menéndez, A.J.L. A multivariate framework for the analysis of the digital divide: Evidence for the European Union-15. *Information & Management* 43, 6 (Sept. 2006), 756–766.
6. Dolan, R.J. and Matthews, J.M. Maximizing the utility of customer product testing: Beta test design and management. *Journal of Product Innovation Management* 10, 4 (Sept. 1993), 318–330.
7. Downey, J.P. and Rainer Jr., R.K. Accurately determining self-efficacy for computer application

domains: An empirical comparison of two methodologies. *Journal of Organizational and End User Computing* 21, 4 (2009), 21–40.

8. Dunahee, M., Lebo, H. et al. *The World Internet Project International Report, Sixth Edition*. University of Southern California Annenberg School Center for the Digital Future, Los Angeles, CA, 2016.
9. Field, A. and Hole, G. *How to Design and Report Experiments*. SAGE Publications, Thousand Oaks, CA, 2002.
10. Hill, T., Smith, N.D., and Mann, M.F. Communicating innovations: Convincing computer-phobics to adopt innovative technologies. *NA-Advances in Consumer Research* 13 (1986), 419–422.
11. Kanij, T., Merkel, R., and Grundy, J. An empirical investigation of personality traits of software testers. In *Proceedings of the IEEE/ACM Eighth International Workshop on Cooperative and Human Aspects of Software Engineering* (Florence, Italy, May 18). IEEE, 2015, 1–7.
12. Malhotra, N.K., Kim, S.S., and Agarwal, J. Internet users' information privacy concerns: The construct, the scale, and a causal model. *Information Systems Research* 15, 4 (Dec. 2004), 336–355.
13. Mäntylä, M.V., Ikonen, J., and Iivonen, J. Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal* 20, 1 (Mar. 2012), 145–172.
14. Merkel, R. and Kanij, T. *Does the Individual Matter in Software Testing?* Technical Report. Centre for Software Analysis and Testing, Swinburne University of Technology, Melbourne, Australia, May 2010.
15. Murphy, C. Where's Japan? *Consumer and Shopper Insights* (Sept. 2011). McKinsey & Company, New York.
16. Ono, H. and Zavodny, M. Digital inequality: A five-country comparison using microdata. *Social Science Research* 36, 3 (Sept. 2007), 1135–1155.
17. Pan, J. Software testing. *Dependable Embedded Systems* 5 (Spring 1999); <https://pdfs.semanticscholar.org/28ab/bfddcd695f6ffc18c5041f8208d0c8810aaf.pdf>
18. PassMark Software. *CPU Benchmarks*; <https://www.cpubenchmark.net/>
19. Perino, J. *6 Different Types of Betabound Testers: Which Are You?* Sept. 11, 2014; <http://www.betabound.com/6-types-beta-testers/>
20. Sauer, J., Seibel, K., and Rüttinger, B. The influence of user expertise and prototype fidelity in usability tests. *Applied Ergonomics* 41, 1 (Jan. 2010), 130–140.
21. Sheehan, K.B. Toward a typology of Internet users and online privacy concerns. *The Information Society* 18, 1 (2002), 21–32.
22. uTest, Inc. *The Future of Beta Testing: 6 Tips for Better Beta Testing*. White Paper. Southborough, MA, Sept. 2012; http://www.informationweek.com/pdf/whitepapers/approved/1376402531_uTest_Whitepaper_Beyond_Beta_Testing.pdf
23. Venkatesh, V., Morris, M.G., Davis, G.B., and Davis, F.D. User acceptance of information technology: Toward a unified view. *MIS Quarterly* 27, 3 (Sept. 2003), 425–478.
24. Wallace, S. and Yu, H.-C. The effect of culture on usability: Comparing the perceptions and performance of Taiwanese and North American MP3 player users. *Journal of Usability Studies* 4, 3 (May 2009), 136–146.

Vlasta Stavova (vlasta.stavova@mail.muni.cz) is a Ph.D. candidate in the Centre for Research on Cryptography and Security in the Faculty of Informatics at Masaryk University, Brno, Czech Republic.

Lenka Dedkova (ldedkova@fss.muni.cz) is a postdoc researcher in the Institute for Research on Children, Youth and Family in the Faculty of Social Sciences at Masaryk University, Brno, Czech Republic.

Martin Ukrop (mukrop@mail.muni.cz) is a Ph.D. candidate in the Centre for Research on Cryptography and Security in the Faculty of Informatics at Masaryk University, Brno, Czech Republic.

Vashek Matyas (matyas@fi.muni.cz) is a professor in the Centre for Research on Cryptography and Security in the Faculty of Informatics at Masaryk University, Brno, Czech Republic.

Copyright held by the authors.
Publication rights licensed to ACM. \$15.00

The challenge of computing in a highly dynamic environment.

BY OTHON MICHAEL AND PAUL G. SPIRAKIS

Elements of the Theory of Dynamic Networks

A DYNAMIC NETWORK is a network that changes with time. Nature, society, and the modern communications landscape abound with examples. Molecular interactions, chemical reactions, social relationships and interactions in human and animal populations, transportation networks, mobile wireless devices, and robot collectives form only a small subset of the systems whose dynamics can be naturally modeled and analyzed by some sort of dynamic network. Though many of these systems have always existed, it was not until recently the need for a formal treatment that would consider *time* as an integral part of the network has been identified. Computer science is leading this major shift, mainly driven by the advent of low-cost wireless communication devices and the development of efficient wireless communication protocols.

The early years of computing could be characterized as the era of staticity and of the relatively predictable; centralized algorithms for (combinatorial optimization) problems concerning static instances, as is that of finding a minimum cost traveling salesman tour in a complete weighted graph, computability questions in cellular automata, and protocols for distributed tasks in a static network. Even when changes were considered, as is the case in fault-tolerant distributed computing, the dynamics were usually sufficiently slow to be handled by conservative approaches, in principle too weak to be useful for highly dynamic systems. An exception is the area of online algorithms, where the input is not known in advance and is instead revealed to the algorithm during its course. Though the original motivation and context of online algorithms is not related to dynamic networks, the existing techniques and body of knowledge of the former may prove very useful in tackling the high unpredictability inherent in the latter.

In contrast, we are rapidly approaching, if not already there, the era of *dynamicity* and of the highly *unpredictable*. According to some latest reports, the number of mobile-only Internet users has already exceeded the number of desktop-only Internet users and

» key insights

- We are rapidly approaching the era of dynamicity and of the highly unpredictable. A great variety of modern networked systems are highly dynamic both in space and time.
- Theory will continue sitting at the center of progress in our science and its necessity toward our understanding of dynamic networks is already evident.
- Many traditional approaches and measures for static networks are not adequate for dynamic networks. There is already strong evidence that there is room for the development of a rich theory.
- Despite the considerable recent progress discussed in this article, we do not yet really know how to compute in highly dynamic environments.



more than 75% of all digital consumers are now using both desktop and mobile platforms to access the Internet. The Internet of Things, envisioning a vast number of objects and devices equipped with a variety of sensors and being connected to the Internet, and smart cities³⁷ are becoming a reality (an indicative example is the recent £40M investment of the U.K. government on these technologies). Computer scientists, nanoscientists, and engineers are joining their forces toward the development of programmable matter, that is, matter that can algorithmically change its physical properties, and have already produced the first impressive outcomes, such as programmed DNA molecules that self-assemble into desired structures¹⁶ and large collectives of tiny identical robots that orchestrate resembling a single multi-robot organism.³⁹ Other ambitious long-term applications include molecular computers, collectives of nanorobots injected into the human circulatory system for monitoring and

treating diseases, or even self-reproducing and self-healing machines. What all of these systems have in common is their characteristic of typically being highly dynamic both in space and time.

The theoretical and analytic approach, prominent in computer science research from the very beginning, has been invaluable in modeling real-world systems and problems, abstracting their essential properties, and answering what can or cannot be done in ideal, extreme, or average conditions. Its findings have constantly enlightened and reshaped applied research and it has revealed some of the deepest and most outstanding models, notions, problems, and theorems of modern mathematics, such as the Turing Machine and Turing's proof on the Entscheidungs problem, the **P** vs. **NP** question, the theory of **NP**-completeness, the four-color theorem, the traveling salesman problem, primality testing, Lamport's causality,²⁷ and the FLP im-

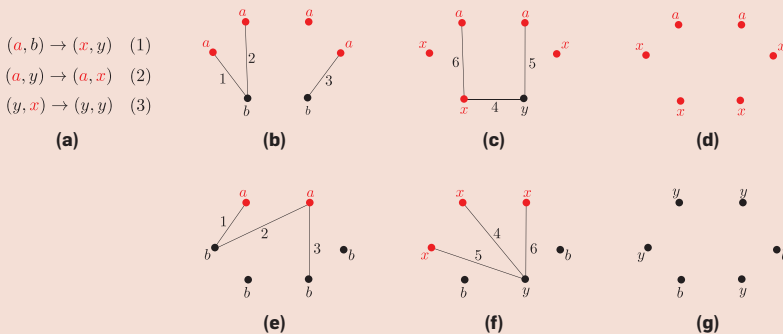
possibility of distributed consensus,²¹ to name just a few.

Theory will continue sitting at the center of progress in our science and its necessity toward our understanding of dynamic networks is already evident. We have reached a point at which a large gap has been formed between existing systems and applications on one side and our fundamental understanding of their underlying principles on the other. Though theory has already identified some first core questions and has provided some preliminary answers to them, it has to run faster in order to bridge the gap and catch up to practice. What computations can be performed by a collection of automata, such as nanodevices or even molecules, which cannot control their own interactions? Even if the computing entities are powerful devices, like smartphones or tablets, can they still carry out basic distributed tasks, such as leader election or counting the size of the system, and with what algorithmic techniques and under what required guarantees about the network's dynamics? Are the traditional network measures adequate for dynamic networks? If not, how can we represent and measure basic quantities, like the speed of information propagation or the diameter, in a network that changes perpetually? Can we continue proving rigorous average-case or even worst-case guarantees and limitations as we have very successfully done for static systems? How can we design a dynamic network that satisfies some desired connectivity properties while minimizing some cost constraints (for example, associated with the fact that creating and maintaining a connection does not come for free)? Which structural and algorithmic properties of static graphs carry over to temporal graphs (an invaluable abstraction of dynamic topologies) and which need a radically new perspective? All these are questions whose ultimate answers are to be provided by the theoreticians.

The existing literature can be roughly partitioned into three clearly distinguishable but also closely interrelated sub-areas: Population protocols, powerful dynamic distributed systems, and temporal graphs. The population protocol model, proposed by Angluin et al. in 2004,⁴ was originally motivated

Figure 1. A population protocol computing whether the number of *a*s in the input is a strict majority.

Initially, each node is in an input-state *a* or *b*. Let N_a and N_b denote the initial number of *a*s and *b*s, respectively. If $N_a > N_b$ we want all nodes to stabilize their output to 1 and to 0 otherwise. (a) The code of the protocol. The possible states are *a*, *x* (red states), *b*, and *y* (black states). The output of red states is 1 and the output of black states is 0. Rule (1) means that when an *a* interacts with a *b*, the former becomes *x* and the latter *y* (similarly for (2) and (3)). To see this protocol is guaranteed to stabilize to a correct output, note that all rules preserve the difference $N_a - N_b$, and consider the last time (1) occurs (when the smaller of *a* or *b* disappears). If $N_a > N_b$, then some nodes remain in state *a*, so (2) and (3) compete to change *y* to *x* and back. However, (2) eventually wins with probability 1, which results in only red states present. This stabilizes the population to output 1, since all rules require a black state to execute. If $N_a \leq N_b$, then rules (1) and (2) are disabled once the final *a* is gone. The last occurrence of (1) ensures at least one *y* exists, so rule (3) then converts all *x*s to *y*s, resulting in only black nodes present, which stabilizes the population to output 0. (b-d) An example execution where $N_a > N_b$. The time-labeled edges indicate in which step the corresponding interaction occurs. (c) After three steps, all *b*s have been eliminated. (d) After another three steps there are only reds and the output 1 is stable. (e-g) An example execution where $N_a < N_b$.



by highly dynamic networks of simple sensor nodes that cannot control their mobility. As the work of Angluin back in 1980³ that is generally accepted as a landmark study for distributed computing in static networks, population protocols could be considered as the starting point of distributed computing in dynamic networks. The other main sub-area originates from the 2005 work of O'Dell and Wattenhofer,³⁸ and later the work of Kuhn, Lynch, and Oshman,²⁵ who reconsidered classical distributed tasks, like leader election, counting, and information dissemination, in a network whose dynamics are captured by a worst-case temporal graph. In parallel, starting from the studies of Berman⁹ and Kempe et al.,²⁴ an increasing number of groups are interested in investigating the structural and algorithmic properties of temporal graphs, which are, roughly speaking, graphs that evolve over time, with the aim at developing a temporal extension of graph theory. In what follows, we will have the opportunity to look deeper into each one of these lines of research. We encourage the interested reader to complement his/her reading of this article with some of the existing technical introductory texts.^{7,11,26,30,32}

Population Protocols: A Soup of Automata

Imagine a population of nanodevices interacting randomly with each other in a well-mixed solution, like a boiling liquid. Each device has a small memory, whose size does not depend on the size of the population, and it has no control over its own mobility, which stems solely from the dynamicity of the environment. The only things that these devices can do, is to obtain an input, for example, by performing a sensing measurement, to update their local state during an interaction with some other device (by applying to their local states a common simple program, called *protocol*, executed by all the devices), and to give an output. An interaction may simply occur when two devices come sufficiently close to each other to establish some sort of communication.

At this point, the reader may be wondering the same that Angluin et al.^{4,6} asked themselves: Can such a soup of automata really compute

anything useful? Angluin et al. proved that actually they can, but not that much, compared to what one is used to expect from modern computing systems. First of all, non-trivial terminating computations are impossible in this model. Indeed, if a node (that is, a device) terminates in some execution, then the same local execution may also result as part of another execution of the same protocol on a larger population, in which case the node terminates and decides without having heard from all the other nodes (imagine a node deciding that there is an even number of nodes with input 1, without knowing all the inputs). One important consequence of this fundamental inability, is that, in this model, we cannot sequentially compose protocols, which makes very challenging the development of protocols for composite tasks. Moreover, we can only hope for computations that *stabilize eventually*, in the sense the nodes always manage to reach a point at which their outputs cannot change any more, even though the nodes cannot actually tell that this has happened. For example, to (stably) compute the parity of the 1s in the whole distributed input, whenever an odd number of nodes have input 1, all nodes must eventually stabilize their output to 1, and to 0, otherwise. This is the parity predicate, which is true if and only if the number of 1s in the input is odd.

But in order to hope for such global computations, we must also say something about the pattern of interactions between the nodes. Imagine, for example, that two parts of the system never influence each other or that some nodes always interact at inconvenient times. In the first case, the system consists of two isolated sub-systems and in the second the environment has the power to enforce some inconvenient symmetries that the protocol cannot break. So, we have to restrict ourselves to environments that are “connected” and “random” enough to not suffer from such inconveniences. There are two main ways to satisfy this: either by assuming the interactions happen in a *fair* manner, essentially meaning they do not forever avoid an always reachable configuration of the system, or they happen uniformly at random from all possible interactions. The former

way is very handy for answering computability questions, while the latter is usually preferred when one wants to analyze the running time of a protocol (that is, the expected number of interactions until stability). See Figure 1 for an example of the model in action.

Angluin et al. managed to give an exact characterization of the computational capabilities of such systems. They proved that if the environment is fair, then the devices can stably compute precisely the *semilinear* predicates, and that this is also true for several interesting variations of the model. But what does this mean exactly? Let us give a simple illustration. Assume that when a device senses its environment, it either sees an a or a b , and denote by N_a and N_b the total number of a s and b s sensed by all the devices, respectively. Then there is a protocol that, on any population and any combination of sensed inputs, can stably compute whether at least $1/3$ of the nodes have seen an a . In other words, the predicate that is true whenever $N_a \geq (N_a + N_b)/3 \iff 2N_a - N_b \geq 0$, is stably computable. The semilinear predicates are precisely those predicates that can be expressed in the form of a linear combination of input variables compared to a constant, that is, $\sum_{i=1}^k \gamma_i N_i < c$ (where the inequality can be of any type, equality inclusive, and can also be replaced by equivalence modulo an integer constant μ). So, for example, the characterization tells us we can stably compute whether the a s are a strict majority (as in Figure 1), whether the b s have been sensed by at least 5% of the nodes, or whether the size of the population is odd. On the other hand, we cannot compute even the simplest expressions involving multiplications of input variables and expressions requiring any form of global iterative sub-computations, such as whether the number of c s is the product of the number of a s and the number of b s or whether the number of b s is a power of 2. The positive part of the characterization is constructive, which means there is a generic protocol that can be adjusted to compute any semilinear predicate.

Now that we know exactly what can be computed in this setting, we may ask: How fast can it be computed? Angluin et al.⁴ proved that if the interactions happen uniformly at random, one at a

time, then the aforementioned generic protocol stabilizes in an expected number of $O(n^2 \log n)$ interactions, where n is the size of the population. Can we do much better than this? Angluin, Aspnes, and Eisenstat⁵ showed that, if there is a pre-elected unique leader in the population, the time of computing any semilinear predicate can be reduced to $O(n \log^5 n)$. A natural next question was whether this speed-up could still be achieved by electing a leader instead of assuming it. Doty and Soloveichik¹⁸ showed recently that it cannot, by proving that an average of $\Omega(n^2)$ interactions have to be paid by any protocol that elects a leader.^a

Despite the indisputable fact that the semilinear predicates constitute a rather small class, this class is by no means trivially achieved. Actually, it can get much worse than semilinear, with apparently gentle additional restrictions. One such, studied by Chen et al.,¹³ is to restrict attention to protocols that never go through a “bottleneck” transition, meaning one that can only occur via an interaction between states that have low counts (constant) in the population. Such protocols have the nice property of always avoiding interactions that have a low probability to occur, and, thus, are slow. Unfortunately, it turns out such protocols cannot count at all, and can only answer existence questions, asking whether a certain symbol is present or not in the input. Another, shows up when one tries to totally avoid the election of a leader, in an attempt to obtain inherently symmetric (that is, parallel) protocols, that do not rely on some global symmetry-breaking process, and, thus, are more efficient and more resilient to faults (for example, a crash failure of a processor). Formally defining what it really means to elect a leader in a distributed system is quite challenging, as it may be achieved implicitly and even sometimes in contrast to a protocol’s intention. To this end, Michail and Spirakis³⁴ defined the *symmetry* of a protocol on a given population and input, as the minimum multiplicity of a state throughout an execution, in which the



We have reached a point at which a large gap has been formed between existing systems and applications on one side and our fundamental understanding of their underlying principles on the other.



environment is as symmetric as possible for this protocol in the given setting. Then they proved there are predicates, like parity, that cannot be computed if we require the symmetry of the protocol to be higher than a constant that depends on the size of the protocol. But enough of those weaknesses; let’s see how minimal additional assumptions can allow the devices to cooperate in order to achieve collective complexity and enable much more powerful computations, in spite of the adversarial nature of the environment.

Beyond Semilinearity

Semilinearity is the price that we pay for minimality: an amorphous system of computational entities that have only constant memory and that cannot infer a bound on the time it takes to hear from all the other entities. Relaxing any of these properties can dramatically increase the computational power. If, for example, the nodes are arranged in a line and the only interactions that can occur are between neighboring nodes in the line, then it is fairly straightforward to simulate a Turing machine of linear space. Similar improvements are possible if the pattern of interactions adheres to some probability distribution. Angluin et al.⁴ showed that, if they happen uniformly at random, then the nodes can simulate a log-space Turing machine with high probability (w.h.p.).

The crucial role of memory in this type of systems has been extensively highlighted and has given some of the most impressive results in this area. One of the restrictions related to local memory that was early questioned, was *anonymity*, that is, the fact that nodes in the original model do not have and cannot ever obtain unique identifiers (ids), simply because there is not enough room in their memory to store them. However, in practice, it is reasonable to expect that even nanodevices will have access to ids, as several existing microcontrollers are set by the factory to store a unique serial number. Guerraoui and Ruppert²³ studied such a variant of the original model, and showed it can simulate a pointer machine, yielding a computational power equal to that of a nondeterministic Turing machine of space $O(n \log n)$.

The effect of explicitly allowing to the devices a larger working memory,

^a Doty and Soloveichik call this a linear-time lower bound, as they perform their analysis in terms of parallel time, simply defined as sequential time divided by n . Both ways are almost equally used in the recent literature. In this article we have chosen to give all bounds in terms of sequential time.

was first studied by Chatzigiannakis et al.¹² Though for theoretical purposes it is quite reasonable to stick to memories that do not scale with the size of the system, this is quite an excessive requirement for real systems. Even for a population as large as 2^{273} nodes, which, by the way, is a number greater than the current estimates of the number of atoms in the observable universe, a logarithmic local memory is for most practical purposes as small as a few hundreds of cells, while most modern micro-controllers come with at least 16KB of RAM. Chatzigiannakis et al. showed that $\Theta(\log \log n)$ local memory is a threshold, under which (asymptotically) semilinearity persists and at which the first non-semilinear predicates become feasible, like computing whether the multiplicity of an input symbol is a power of 2. They also proved that if the local memories have size $f(n) = \Omega(\log n)$, then the computational power is equivalent to that of a nondeterministic Turing machine of space $O(nf(n))$ and there is a space hierarchy, essentially meaning that protocols having access to more memory can compute more things.

If a moderate increase of local memory is additionally combined with a guarantee of a uniformly random interaction pattern, then even more fascinating tasks become feasible. Michail²⁹ showed that, in this case, a pre-elected unique leader with two n -counters can terminate and still count an upper bound on the size n of the system w.h.p. The idea is to have the leader implement two competing processes, running in parallel. The first process counts the number of nodes that have been encountered once and the second process counts the number of nodes that have been encountered twice. The game ends when the second counter catches up the first. It can be proved that when this occurs, the leader will almost surely have already counted at least half of the nodes. Alistarh and Gelashvili² showed that $O(\log \log n)$ bits of memory per node are sufficient to elect a unique leader in an expected number of $O(n \log^3 n)$ interactions, a great improvement compared to the $\Omega(n^2)$ lower bound for the constant-memory case.

Natural Processes and Programmable Matter

Apart from being a model of computing in a highly dynamic environment, population protocols bear some striking

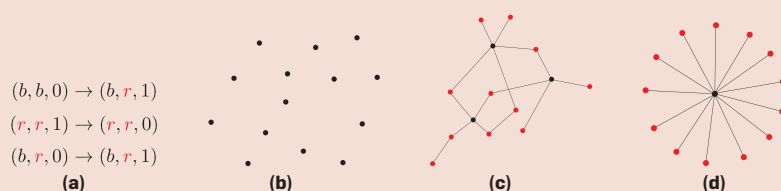
similarities to several natural processes. They can be viewed as an abstraction of “fast-mixing” physical systems, such as chemical reaction networks, animal populations, and gene regulatory networks. The strong resemblance of population protocols to models of interacting molecules in theoretical chemistry had already been observed by Angluin et al.⁴ Assuming a fixed molecular population size and bi-molecular reactions, population protocols are formally equivalent to chemical reaction networks, a formal model of chemistry in a well-mixed solution, describing how certain species of molecules within a solution, such as DNA strands, react to produce new species.¹⁷ An important consequence of this, is that bounds and characterizations for population protocols, apart from being useful for computer science applications, usually translate to inherent properties of natural systems. For example, the “molecular translation” of the aforementioned $\Omega(n^2)$ lower bound for leader election,¹⁸ is that it is essentially difficult to generate exact quantities of molecular species quickly (at least slower than destroying all molecules of the species, which takes $O(n \log n)$ time). There are even population protocols, like those for computing an approximate majority, that have been connected to biological networks.¹⁰ Czyzowicz et al.¹⁵ have recently studied the relation of population protocols to antagonism of species, with dynamics modeled by discrete Lotka-Volterra equations.

Another interesting possibility recently highlighted by Michail and Spirakis is to use population protocols as a model of (algorithmic) distributed net-

work construction and, consequently, as a potential model for programmable matter able to self-organize in a dynamic environment. Michail and Spirakis³⁵ studied an extension of population protocols, called network constructors, in which the devices can additionally establish bonds with each other (like a molecular bonding mechanism);^b see Figure 2 for an example. One of their main results was that such systems can construct as complex stable networks as those that can be decided by a centralized algorithm. The idea is to program the nodes to organize themselves into a network that can serve as a memory of size $O(n^2)$, which is asymptotically maximum and can only be achieved by exploiting the presence or absence of bonds between nodes as the bits of the memory (if only the nodes' local space was used, then the total memory could not exceed $O(n)$). Then the population draws a random network and simulates on the distributed memory a Turing machine that decides whether the network belongs to the target ones. If yes, the population stabilizes to it, otherwise the random experiment and the simulation are repeated. What makes the construction intricate is that all the sub-routines have to be executed in parallel and potential errors due to this to be corrected by global resets throughout the course of the protocol. Michail²⁹ then studied a more applied version of this model, by adding geometric constraints (representing physical restrictions), according to which the formed network and

^b A predecessor of this model had served as one of the first computationally powerful variants of population protocols, exploiting bond states to simulate an $O(n^2)$ -space nondeterministic Turing machine.³¹

Figure 2. (a) A simple optimal protocol that allows the nodes to self-organize into a global star. Blacks eliminate each other, reds repel, and blacks attract reds. (b) Initially all nodes are black and no active connections exist. (c) After a while, only three blacks have survived each having a set of red neighbors. (d) A unique black has survived, it has attracted all reds, and all connections between reds have been deactivated. The construction is a stable global star.



the allowable interactions must respect the structure of the two-dimensional (or three-dimensional) grid network.

Powerful Dynamic Distributed Systems

As we have seen, population protocols and their variants concern some rather specialized computing systems operating in fairly extreme conditions. Typical dynamic distributed systems, usually consist of much more “gifted” devices, like smartphones or tablets, equipped, among other things, with ids, practically unbounded local memories, powerful processors, and wireless (radio) transceivers.

One of the first formal models for such systems, developed by O’Dell and Wattenhofer,³⁸ appeared in 2005, just one year after the original paper of Angluin et al. on population protocols. Their model is, essentially, a generalization of classical networked message-passing distributed systems, where, instead of a static graph, the underlying network is now represented by an unknown and adversarially controlled (that is, worst-case) temporal graph.^c The nodes are Turing machines with unbounded tapes (it is the protocol designer’s responsibility to minimize both the actual space used and the local processing time) that communicate with other nodes by interchanging messages over a wireless medium. As is always the case, the worst-case approach has the benefit that the results hold for all possible dynamic network topologies (of course, between those that make sense) and not just for some convenient special cases or distributions. In order to allow for bounded end-to-end communication, O’Dell and Wattenhofer imposed on the underlying dynamic network the restriction of being connected at any instant.^d Such a simplification may sound artificial, as most real dynamic systems are expected to almost never be connected, still it is very convenient for the

purpose of theoretical analysis and for establishing some first fundamental principles. We will discuss more recent studies that have developed ways of relaxing this restriction.

O’Dell and Wattenhofer defined their model in terms of asynchronous communication and studied the token dissemination and routing problems in this setting. In token dissemination, a token, that is, a piece of information, is initially present on some source node and the goal is to distribute the token to the entire network and have all nodes terminate when dissemination has successfully completed (for example, when a base station wants to disseminate to all nodes in a sensor network a global reset signal). In routing, the token has only to be delivered to a designated destination node.

Five years later, Kuhn, Lynch, and Oshman²⁵ proposed a synchronous version of the model above, which substantially simplified thinking and treating dynamic networks formally, and, thus, lead to numerous new insights and directions. The nodes operate now in lock-step, synchronized in discrete rounds either by having access to a global clock or by keeping local clocks synchronized. In every round, an adversary scheduler (modeling the worst-case nature of the network’s dynamicity) selects a set of edges between the nodes and every node may communicate with its current neighbors, as selected by the adversary, usually by broadcasting a single message to be delivered to all its neighbors. As in the previous model, the network is revealed to the distributed algorithms in an online and totally unpredictable way and the nodes have no a priori knowledge about the network apart from the guarantee that its instances are connected. Despite the simplicity of the model, even the most basic distributed tasks no longer seem straightforward. For example, how can the nodes count the size n of the system and terminate (that is, be able to detect that their task has successfully come to an end)?

To appreciate the difficulties, it is useful to see why a typical approach for static networks fails in dynamic networks. In static networks, the stability of paths is an invaluable implicit guarantee for the rate of global progress. In particular, if a node u broadcasts a message and every node that receives the

message forwards it to all its neighbors, then u knows that, in every round, at least one more node receives the message for the first time. Moreover, if every node acknowledges the receipt by broadcasting an ack message containing its id, then if all nodes forward these acks, u knows that, in every second round, it must either hear from a new remote node or all nodes must have already received u ’s message. The first guarantee is still satisfied in the dynamic case, because if all nodes that have a piece of information broadcast it in every round, then connectivity of the instantaneous topology ensures that at least one of them will deliver it to a node that has not heard of it yet. However, the same is not true for the second guarantee. Imagine a star topology, with u lying on the center and being directly connected to all other nodes (the peripherals), apart from one node v that is connected to a peripheral node w but not directly to u . In round 1, u can learn about the existence of all nodes but v . Then, in round 2, the topology changes to a line spanning the nodes, with u lying on the left endpoint, v on the right, and w being the unique neighbor of v in the line, and remains static forever. As the only nodes that know about v are the two rightmost nodes of the line, it will take $n-2$ more rounds for u to realize that another node exists. Given that u does not know any estimate of n in advance, at first sight it seems that u has no means of determining how long it should wait. The good news is it is still possible to *infer* such a bound.

Before showing how, let us first extract from this discussion two very useful notions for capturing the spread of influence in a dynamic distributed system. Both are based on Lamport’s *causal influence*,²⁷ which formalizes the notion of one node “influencing” another through a chain of messages (possibly going through other nodes in between). The first one is the *future set* of a node u in a given time interval, containing all nodes that u has influenced in that interval. The second one is the *past set* of a node u in a given time interval, containing all nodes that have influenced u in the interval. Stated in the new terminology, this discussion says that the cardinality of u ’s future set increases by at least one in every round, until it becomes equal to n . Though, as we have highlighted, the same is not true for the

^c One way to define a temporal graph D is as a pair (V, A) , where V is a static set of nodes and $A: \mathbb{N} \rightarrow \mathcal{C}(V)$ a mapping, such that $A(t)$ is the (possibly empty) set of all edges that appear at time t (time-edges). Then a *temporal path* or journey of D is a path of time-edges using increasing times.

^d If the temporal graph of the dynamic network is $D = (V, A)$, then this means that, for all times $t \in \mathbb{N}$ the static graph $G = (V, A(t))$ is connected.

past set, still there is an alternative and equally useful guarantee on its rate of growth. The size of the past set of u is an upper bound on the number of rounds required for u to hear of a new node, that is, for its past set to increase by at least one. This is because the set of nodes that know a new influence for u are initially those nodes not in the past set of u and, due to connectivity, in every round the former set increases by at least one, so in a number of rounds at most equal to the size of u 's past set the whole past set will know a new influence, u inclusive.

By a simple induction on the number of rounds, we obtain that the size of u 's past set must be either greater than the number of the current round or equal to n . This immediately gives to u a way for knowing when it has heard of all nodes: keep track of your past set in a list A (for example, recording nodes' ids) and of the current round r ; if it ever holds that $r \geq |A|$, then A contains all nodes in the system. This idea gives an $O(n)$ -round distributed algorithm for counting the size of the system, and, by changing the output and the contents of the transmitted messages, is also an algorithm for many other basic distributed tasks, such as information dissemination, leader election, and computing arbitrary functions on inputs to the nodes. For these algorithms to work, all nodes must broadcast in every round all information that they know, which is not a desired property as it results in transmitting very large messages, that is of size $O(n \log n)$.

Kuhn, Lynch, and Oshman also developed an alternative approach that uses only $O(\log n)$ bits per message, a much more reasonable message overhead for real systems, paying a linear factor increase in termination time. The idea is as follows. The nodes have a guess k of the size of the system and then try to verify whether their guess was a correct upper bound on n . If it was, then it is possible for the nodes to terminate knowing the exact value of n , otherwise they double k and repeat. Assume, for simplicity, a unique leader coordinates the verification process. What the leader does is to invite $k-1$ other nodes to join its committee. Each node that is not invited creates its own committee. As long as the guess is not correct, there must be at least two committees, and when it becomes correct for the first time, there will be a sin-



We are on the road to a unified theory of dynamic networks, but not there yet.



gle committee (that is, the leader's) containing all nodes. Then it is fairly simple for the nodes to verify whether there is precisely one committee. Having a leader coordinate the process is crucial for reducing the message overhead, as, in this way, the other nodes need only broadcast the information emanating from the leader. Fortunately, the leader need not be assumed, but can be elected in parallel with this process, without increasing the size of the messages.

As noted earlier, continuous connectivity was one of the first assumptions of these models to be questioned. Michail, Chatzigiannakis, and Spirakis³³ replaced it by more general conditions of *temporal connectivity*, that is, connectivity satisfied over time. To do this, they introduced metrics to capture the speed of influence propagation in networks that are possibly disconnected at all times. These metrics concern properties that do not necessarily hold in every round, but instead may require several rounds until they are satisfied. One such is the *connectivity time* of a dynamic network, which is the maximal time that the two parts of any cut of the network can remain disconnected. Another is the *outgoing influence time*, which is the maximal time until the state of a node at a given time (for example, its initial state) in the state of another node. They gave efficient distributed algorithms for counting and information dissemination, by exploiting a known upper bound on each of these metrics.

The *temporal diameter*, a measure of the time required for influence dissemination, generalizes the standard network diameter, as the latter is unsuitable for dynamic networks. It is defined as the minimum integer d for which it holds that the *temporal distance* (that is, the duration of a journey of minimum arrival time) between every ordered pair of nodes at any given time is at most d . For an indicative example, consider a dynamic star in which all peripherals ($u_1, u_2, \dots, u_{(n-2)}$) but two ($u_{(n-1)}, u_n$) go to the center one after the other in a modular way; that is, at any time $t \geq 0$, $u_{[t \bmod (n-2)]+1}$ is the center of the star and all the other nodes are peripherals. Then any message from u_{n-1} to u_n needs $n-1$ steps to be delivered, because u_n can only get the message if a node that has already obtained it becomes the center

again. So, the temporal diameter of this network is $n-1$ even though its instantaneous diameter is at any given time just 2. This is a simplified version of a construction used by Avin, Koucký, and Lotker⁸ to show that, in contrast to the cover time of a random walk on a static graph, which is always polynomial in n , the cover time of a random walk on a temporal graph may be exponential. The diameter is just one of those many network notions that have to be redefined to take time into account, in order to become suitable for dynamic networks.

In practice, the network dynamics may not always be totally unpredictable or irregular. The dynamicity patterns of many real-world systems, such as human interactions and transportation units, exhibit regularities and are to some extent predictable. In view of this, some authors considered network dynamics that are a result of randomness, while others deterministic network dynamics that are recurrent or periodic. Clementi et al.¹⁴ studied the speed of information dissemination in the following type of edge-Markovian dynamic networks: if an edge exists at time t then, at time $t+1$, it disappears with probability q , and if instead the edge does not exist at time t , then it appears at time $t+1$ with probability p . Flocchini, Mans, and Santoro²² studied one type of periodic dynamic networks, called carrier networks, in which the dynamic network is defined by the periodic movements of some mobile entities, called carriers. This is a natural abstraction of several real-world systems like public transports with fixed timetables, low earth orbiting satellite systems, and security guards' tours. They studied the problem of exploring all nodes of the network by an agent who can only follow the route of a carrier (like a passenger) and can switch from one carrier to another.

Structural Properties of Temporal Graphs

Modern dynamic systems and applications, as well as the theoretical progress in dynamic distributed systems described so far, led several researchers to the realization that the underlying topology model of dynamic networks is not a mere generalization of graphs; rather, it manifests some essentially different structural and algorithmic properties. A temporal extension of graph

theory is already under development, with the aim at delivering a concrete set of results, tools, and techniques for temporal graphs. Graphs have proved to be an invaluable tool for representing and enabling the formal treatment of relatively stable networked systems. There are already strong indications that temporal graphs will play an equally important role for dynamic networks.

A temporal graph can be thought of as a special case of labeled graphs, where labels capture some measure of time, for example, the precise times or time intervals at which each connection is available. But is there anything new here? Can't we just resort to traditional graph approaches to deal with this seemingly minor extension? A first indication that the answer might not be that obvious, is the richness that emerged from considering labels as colors and trying to solve conflict-free coloring problems (strongly motivated by real-world problems, like frequency assignment in cellular networks), in the classical and well-studied area of graph coloring. Indeed, the main message from existing research on temporal graphs is that many graph properties and problems become radically different and usually substantially more difficult when an extra time dimension is added to them. This was first highlighted by Kempe, Kleinberg, and Kumar²⁴ in a minimal special case of temporal graphs, in which every edge is available only once. They proved that, in such temporal graphs, the classical formulation of Menger's theorem^e is violated if applied to journeys and the computation of the number of node-disjoint s - z paths becomes **NP**-complete. A reformulation of Menger's theorem which is valid for all temporal graphs was recently achieved by Mertzios et al.²⁸

The algorithmic problems of temporal graphs can be divided into two main types, depending on the algorithm's knowledge about the future evolution of the graph. Online algorithms have no knowledge about the future, while offline algorithms know the full evolution of the graph in advance.

^e Menger's theorem states the maximum number of node-disjoint s - z paths is equal to the minimum number of nodes whose removal separates node s from node z .

An example of an online centralized problem on temporal graphs is *k-token dissemination*, which asks to disseminate to all nodes as fast as possible, k tokens that are initially assigned to some of the nodes. The only restriction on the algorithm's knowledge is that it has to make its selection of tokens to be forwarded without knowing the edges selected by the adversary in the current round. Kuhn, Lynch, and Oshman²⁵ showed by a potential function argument that any such deterministic centralized algorithm for the problem in continuously connected temporal graphs requires at least $\Omega(n \log k)$ rounds to complete in the worst case (their corresponding distributed upper bound, by the algorithm described in the previous section, is $O(nk)$). This lower bound was further improved to $\Omega(nk/\log n)$ by Dutta et al.¹⁹ via the probabilistic method.

Even though offline centralized algorithms do not suffer from the unpredictability that characterizes dynamic network problems, still offline temporal versions of standard graph problems can be substantially more difficult to solve.

One such example, studied by Michail and Spirakis,³⁶ is the problem of exploring the nodes of a temporal graph as soon as possible. Though, in the static case, the decision version of the problem can be solved in linear time and the optimization version (known as GRAPHIC TSP) can be satisfactorily approximated, in the temporal case the decision version becomes **NP**-complete and there exists some constant $c > 0$ such that the optimum solution cannot be approximated within cn (meaning that we cannot find a solution, which is at most cn times worse than the optimum), unless **P** = **NP** (things do not become any better, even if all instances are connected^{20,36}).

Mertzios et al.²⁸ also studied the problem of designing a cost-efficient temporal graph, given some requirements that the graph should meet. Briefly, we are provided with an underlying graph G and we are asked to assign labels to its edges so that the resulting temporal graph minimizes some parameter (related to the cost of making an edge available) while satisfying some connectivity constraint. Other authors have considered random temporal graphs, a succinctly representable model, in

which the labels are chosen according to some probability distribution.¹

The Future

Do we really know how to compute in highly dynamic environments, how to represent and measure their core properties, or even how to efficiently solve centralized computational problems concerning them? Despite the considerable recent progress that we discussed in this article, the answer is: *not yet*. We are on the road to a unified theory of dynamic networks, but not yet there.

Though it is still quite early to anticipate the full range of potential applications, there is already strong evidence that there is room for the development of a rich theory. As is always the case, the groundwork will be laid by our ability to identify and formulate radically new problems and not just by studying adjusted versions of existing ones. Real dynamic systems are the natural place to look for such problems. Still, the existing literature has already identified some first challenging research directions and technical problems whose further investigation has the potential to push forward the area.

First, is there a general rule underlying the complexity increase of a network problem when that problem is extended in time? Moreover, most natural applications require an algorithm to operate on a dynamic network without knowing or being able to accurately predict the future evolution of the network. It might be the case that the right treatment of such settings is via online algorithms and analysis, however little effort has been devoted to this. We saw that there is a natural reformulation of Menger's theorem for temporal graphs. It would be very valuable to check the validity of many other fundamental results of graph theory, like Kuratowski's planarity theorem or Mantel's beautiful theorem on the existence of triangles.

Another critical issue has to do with a long-standing problem in distributed computing theory: there is practically a different model for each setting and usually slight modifications of a model result in totally different formal properties. This multiplicity is expected to be even more intense in dynamic networks, due to the almost inexhaustible variety of different dynamicity patterns.

Therefore, if possible, a unification of models for distributed computing in dynamic networks would be more than valuable. There is also a great need for progress in programming and verification of programmable matter protocols, probably the only sub-area of dynamic networks in which theory has grown faster than systems. We need many more real collectives of tiny devices, in order to identify, which assumptions actually make sense and are worth studying and which would never show up in a real system and have to be abandoned. Finally, we should seriously take into account the recent advances in learning and statistical learning in particular, as a rich source of powerful methods for a system to learn and, thus, be able to predict to some extent the pattern of the dynamic network and to adapt its algorithms in order to cope with the new conditions more effectively. 

References

- Akrida, E.C., Gasieniec, L.G., Mertzios, G.B. and Spirakis, P.G. Ephemeral networks with random availability of links: The case of fast networks. *J. Parallel Distrib. Comput.* 87 (2016), 109–120.
- Alistarh, D. and Gelashvili, R. Polylogarithmic-time leader election in population protocols. In *Proceedings of ICALP*, 2015, 479–491.
- Angluin, D. Local and global properties in networks of processors. In *Proceedings of STOC*, 1980, 82–93.
- Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J. and Peralta, R. Computation in networks of passively mobile finite-state sensors. In *Proceedings of PODC*, 2004, 290–299. Also in *Distributed Computing*, 2006.
- Angluin, D., Aspnes, J. and Eisenstat, D. Fast computation by population protocols with a leader. *Distributed Computing* 21, 3 (2008), 183–199.
- Angluin, D., Aspnes, J., Eisenstat, D. and Ruppert, E. The computational power of population protocols. *Distributed Computing* 20, 4 (2007), 279–304.
- Aspnes, J. and Ruppert, E. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, 2009, 97–120.
- Avin, C., Koucký, M. and Lotker, Z. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of ICALP*, 2008, 121–132.
- Berman, K.A. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks* 28, 3 (1996), 125–134.
- Cardelli, L. Morphisms of reaction networks that couple structure to function. *BMC Systems Biology* 8, 1 (2014), 84.
- Casteigts, A., Flocchini, P., Quattrociocchi, W. and Santoro, N. Time-varying graphs and dynamic networks. *IJPEDS* 27, 5 (2012), 387–4082.
- Chatzigiannakis, I., Michail, O., Nikolaou, S., Pavlogiannis, A. and Spirakis, P.G. Passively mobile communicating machines that use restricted space. *Theor. Comput. Sci.* 412, 46 (2011), 6469–6483.
- Chen, H.-L., Cummings, R., Doty, D. and Soloveichik, D. Speed faults in computation by chemical reaction networks. In *Proceedings of DISC*, 2014, 16–30. Also in *Distributed Computing*, 2015.
- Clementi, A. E., Macci, C., Monti, A., Pasquale, F. and Silvestri, R. Flooding time in edge-markovian dynamic graphs. In *Proceedings of PODC*, 2008, 213–222.
- Czyzowicz, J. et al. On convergence and threshold properties of discrete lotka-volterra population protocols. In *Proceedings of ICALP*, 2015, 393–405.
- Doty, D. Theory of algorithmic self-assembly. *Commun. ACM* 55, 12 (Dec. 2012), 88.
- Doty, D. Timing in chemical reaction networks. In *Proceedings of SODA*, 2014, 772–784.
- Doty, D. and Soloveichik, D. Stable leader election in population protocols requires linear time. In *Proceedings of DISC*, 2015, 602–616.
- Dutta, C., Pandurangan, G., Rajaraman, R., Sun, Z. and Viola, E. On the complexity of information spreading in dynamic networks. In *Proceedings of SODA*, 2013, 717–736.
- Erlebach, T., Hoffmann, M. and Kammer, F. On temporal graph exploration. In *Proceedings of ICALP*, 2015, 444–455.
- Fischer, M.J., Lynch, N.A. and Paterson, M.S. Impossibility of distributed consensus with one faulty process. *J. ACM* 32, 2 (1985) 374–382.
- Flocchini, P., Mans, B. and Santoro, N. On the exploration of time-varying networks. *Theoretical Computer Science* 469 (2013), 53–68.
- Guerraoui, R. and Ruppert, E. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *Proceedings of ICALP*, 2009, 484–495.
- Kempe, D., Kleinberg, J. and Kumar, A. Connectivity and inference problems for temporal networks. In *Proceedings of STOC*, 2000, 504–513.
- Kuhn, F., Lynch, N. and Oshman, R. Distributed computation in dynamic networks. In *Proceedings of STOC*, 2010, 513–522.
- Kuhn, F. and Oshman, R. Dynamic networks: models and algorithms. *SIGACT News*, 2011, 42:82–96.
- Lamport, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (July 1978), 558–565.
- Mertzios, G.B., Michail, O., Chatzigiannakis, I. and Spirakis, P.G. Temporal network optimization subject to connectivity constraints. In *Proceedings of ICALP*, 2013, 657–668.
- Michail, O. Terminating distributed construction of shapes and patterns in a fair solution of automata. In *Proceedings of PODC*, 2015, 37–46. Also in *Distributed Computing*, 2017.
- Michail, O. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics* 12, 4 (2016), 239–280.
- Michail, O., Chatzigiannakis, I. and Spirakis, P.G. Mediated population protocols. *Theoretical Computer Science* 412, 22 (2011), 2434–2450.
- Michail, O., Chatzigiannakis, I. and Spirakis, P.G. *New Models for Population Protocols*. N.A. Lynch, ed. Morgan & Claypool, 2011.
- Michail, O., Chatzigiannakis, I. and Spirakis, P.G. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *J. Parallel Distrib. Comput.* 74, 1 (2014), 2016–2026.
- Michail, O. and Spirakis, P.G. How many cooks spoil the soup? In *Proceedings of SIROCCO*, 2016, 3–18.
- Michail, O. and Spirakis, P.G. Simple and efficient local codes for distributed stable network construction. *Distributed Computing* 29, 3 (2016), 207–237.
- Michail, O. and Spirakis, P.G. Traveling salesman problems in temporal graphs. *Theoretical Computer Science* 634, 2016, 1–23.
- Mone, G. The new smart cities. *Commun. ACM* 58, 7 (July 2015), 20–21.
- O'Dell, R. and Wattenhofer, R. Information dissemination in highly dynamic graphs. In *Proceedings of DIALM-POMC*, 2005, 104–110.
- Rubenstein, M., Cornejo, A. and Nagpal, R. Programmable self-assembly in a thousand-robot swarm. *Science* 345, 6198 (2014), 795–799.

Othon Michail (Othon.Michail@liverpool.ac.uk) is a lecturer in the Department of Computer Science at the University of Liverpool, U.K.

Paul G. Spirakis (P.Spirakis@liverpool.ac.uk) is a professor in the Department of Computer Science at the University of Liverpool, U.K. and senior researcher at the Computer Technology Institute, Patras, Greece.

Copyright held by authors/owners.
Publication rights licensed to ACM. \$15.00.



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/elements-of-the-theory-of-dynamic-networks>

research highlights

P. 83

**Technical
Perspective
Building Bug-Free
Compilers**

By Steve Zdancewic

P. 84

**Practical Verification
of Peephole Optimizations
with Alive**

By Nuno P. Lopes, David Menendez,
Santosh Nagarakatte, and John Regehr

P. 92

**Technical
Perspective
Designing
Algorithms and
the Fairness Criteria
They Should Satisfy**

By Vincent Conitzer

P. 93

**Which Is the Fairest
(Rent Division) of Them All?**

By Kobi Gal, Ariel D. Procaccia, Moshe Mash, and Yair Zick

Technical Perspective

Building Bug-Free Compilers

By Steve Zdancewic

SOFTWARE ENGINEERS—AND, by extension, anyone who uses the software they create (that is, nearly everyone)—rely critically on compilers. These ubiquitous tools, familiar even to the most novice programmer, translate high-level ideas, expressed as code, into the low-level instructions understood by computer hardware.

Using compilers is so commonplace and transparent that most of us soon forget about the near-miracles of code analysis and optimization they perform at a keystroke. Today's compilers produce marvelously tuned and optimized code that almost always performs better than what could be achieved painstakingly by hand.

Optimizing compilers are an essential part of our computing infrastructure.

To achieve these near-miracles, compilers themselves are very complex beasts, comprising tens- or hundreds-of-thousands of lines of code, developed over many years by many people. And, just as with any software, they too are potentially buggy. A broken compiler might produce erroneous machine code, thereby converting the programmer's intended algorithm into gibberish, or, worse, into a subtly flawed product that sometimes gets the wrong answer or has a pitfall waiting for just the right input to be triggered.

Fortunately, thanks to their frequent use, compilers tend to be very well-tested software: they do their job correctly most of the time for most programs. Nevertheless, compiler bugs do exist—as demonstrated by Regehr and his collaborators in prior work using their Csmith tool. They can often be provoked by surprisingly small (but tricky!) programs that involve the corner-cases of fixed-bitwidth arithmetic calculations, which are incorrectly optimized when generating low-level code. These kinds of compiler bugs are potentially catastrophic, notoriously difficult for programmers to diagnose, and require real expertise to track down and fix.

So, what do we do about them?

The authors of the following paper give us a compelling and practical answer. Their main idea is to raise the level of abstraction used by compiler implementors, allowing them to express optimizations by using a domain-specific language suited perfectly for the task.

The authors explain their ideas in the context of the LLVM intermediate representation, which, owing to the widespread use of LLVM in both academia and industry (most prominently by Apple), makes the results in this paper immediately applicable—indeed the authors found numerous bugs in the LLVM implementation, and have been working with LLVM developers to find more.

The paper focuses on peephole optimizations, in which the compiler rewrites a short sequence of instructions into a more efficient sequence, for instance, by replacing multiplication by bitshift and addition operations. The traditional way to implement this kind of thing in LLVM is for the compiler implementor to manually write a bunch of C++ code that case-analyzes the LLVM instructions looking for a specific pattern that should be replaced.

Why is it difficult for compiler implementors to get this right? At a superficial level, such code consists of a lot of fiddly boilerplate, and, when there are hundreds of peephole optimizations, as in an industrial-strength compiler like LLVM, bugs are bound to creep in.

But the problem is actually much deeper. One difficulty is the semantics for even “simple” arithmetic operations is not so straightforward. These operations include many corner cases: underflows, overflows, and undefined behaviors, all of which must be properly accounted for to ensure an optimization is correct. Moreover, not all optimizations apply in all situations. The compiler might need to know facts about the program (for

example, that a variable is nonzero) to ensure an optimization applies. When hand-implementing an optimization, the programmer must carefully follow all of these constraints.

The paper describes Alive, a tool that realizes the domain-specific language for describing peephole optimizations. Instead of writing an optimization by hand, the compiler writer expresses it using natural LLVM syntax. Alive generates a C++ implementation, thereby eliminating the ugly boilerplate. More importantly, Alive also verifies the proposed optimization is indeed correct: the tool understands the semantics of LLVM (including the tricky corner cases), and checks that the optimized code agrees with the original. When an optimization is broken, Alive produces a counterexample showing the problem. The tool can even infer parts of the optimization, which lets compiler writers be more aggressive about producing good code, without the fear of bugs.

To show that Alive works in practice, the authors reimplemented hundreds of peephole optimizations from the LLVM suite and stress-tested them to look for miscompilation bugs. They found none.

This paper is not the end of the story for compiler correctness—there are other kinds of optimizations, analyses, and transformations not covered here—but it does provide a large step forward by showing how to build a tool that is practical yet backed by modern verification technology. In short, this paper suggests a way to make better compilers that are easier to build and far less buggy than they are today. 

Steve Zdancewic is a professor in the Department of Computer Science and Information Science at the University of Pennsylvania, Philadelphia, PA, USA.

Copyright held by author.

Practical Verification of Peephole Optimizations with Alive

By Nuno P. Lopes, David Menendez, Santosh Nagarakatte, and John Regehr

Abstract

Compilers should not miscompile. Peephole optimizations, which perform local rewriting of the input program to improve the efficiency of generated code, are a persistent source of compiler bugs. We created Alive, a domain-specific language for writing optimizations and for automatically either proving them correct or else generating counterexamples. Furthermore, Alive can be automatically translated into C++ code that is suitable for inclusion in an LLVM optimization pass. Alive is based on an attempt to balance usability and formal methods; for example, it captures—but largely hides—the detailed semantics of the various kinds of undefined behavior. Alive has found numerous bugs in the LLVM compiler and is being used by LLVM developers.

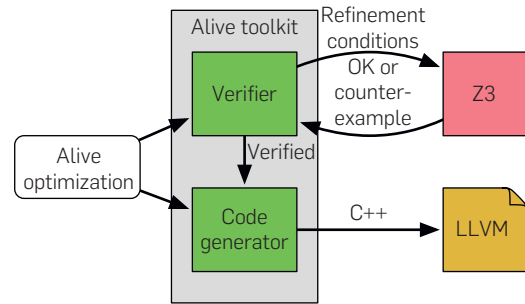
1. INTRODUCTION

Compiler optimizations should be efficient, effective, and correct—but meeting all of these goals is difficult. In practice, whereas efficiency and effectiveness are relatively easy to quantify, correctness is not. Incorrect compiler optimizations can remain latent for long periods of time; the resulting problems are subtle and difficult to diagnose since the incorrectness is introduced at a level of abstraction lower than the one where software developers typically work.

Random testing^{7, 20} is one approach to improving the correctness of compilers; it has been shown to be effective, but of course testing misses bugs. A stronger form of insurance against compiler bugs can be provided by a proof that the compiler is correct (compiler verification) or a proof that a particular compilation was correct (translation validation). The state of the art in compiler verification requires a fresh compiler implementation and many person-years of proof engineering (e.g., CompCert⁹), making this approach impractical in most production environments.

We developed Alive: a new language and tool for developing correct peephole optimizations as shown in Figure 1. Peephole optimizations in LLVM are performed by the instruction combiner (InstCombine) pass. Alive aims for a design point that is both practical and formal; it allows compiler writers to specify peephole optimizations for LLVM's Intermediate Representation (IR), it automatically proves them correct with the help of a Satisfiability Modulo Theory (SMT) solver (or provides a counterexample), and it automatically generates C++ code that is similar to handwritten peephole optimizations such as those found in the instruction combiner. Alive's main contributions are in identifying a subset of peephole optimizations that can be automatically verified and in providing a usable formal methods tool

Figure 1. Overview of Alive. Optimizations expressed in Alive are automatically verified using the Z3 SMT solver. Verified optimizations are converted to C++ implementations for use in LLVM.



based on the semantics of LLVM IR, with support for automated correctness proofs in the presence of LLVM's undefined behavior, and with support for code generation.

InstCombine transformations perform numerous algebraic simplifications that improve efficiency, enable other optimizations, and canonicalize LLVM code. InstCombine optimizations have been a persistent source of LLVM bugs.^{7, 20}

An example InstCombine transformation takes $(x \oplus -1) + C$ and turns it into $(C - 1) - x$ where x is a variable, $\oplus$ is exclusive or, and C is an arbitrary constant as wide as x . If C is 3333, the LLVM input to this InstCombine transformation would look like this:

```
%1 = xor i32 %x, -1
%2 = add i32 %1, 3333
```

and the optimized code:

```
%2 = sub i32 3332, %x
```

In Alive the same optimization is:

```
%1 = xor %x, -1
%2 = add %1, C
=>
%2 = sub C-1, %x
```

The Alive specification is designed to resemble—both syntactically and semantically—the LLVM transformation that it describes. It is much more succinct than its

The original version of this paper is titled “Provably Correct Peephole Optimizations with Alive” and was published in the proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2015.

equivalent C++ implementation, is not expressed in terms of LLVM's internal data structures and control flow, and can be automatically verified by the Alive tool kit. This transformation illustrates 2 forms of abstraction supported by Alive: abstraction over choice of a compile-time constant and abstraction over bitwidth.

So far Alive has helped us discover twenty-three previously unknown bugs in the LLVM InstCombine transformations. Furthermore, we have prevented dozens of bugs from getting into LLVM by monitoring the various InstCombine patches as they were committed to the LLVM subversion repository. Several LLVM developers are currently using the Alive prototype to check their InstCombine transformations. Alive is open source and it is also available on-line at <http://rise4fun.com/Alive>.

2. THE ALIVE LANGUAGE^a

We designed Alive to resemble the LLVM IR because our users—the LLVM developers—are already experts with it. Alive's most important features include its abstraction over choice of constants, over the bitwidths of operands (Section 2.2), and over LLVM's instruction attributes that control undefined behavior (Section 2.4).

2.1. Syntax

An Alive transformation has the form $A \Rightarrow B$, where A is the *source template* (unoptimized code) and B is the *target template* (optimized code). Additionally, a transformation may include a precondition. Since Alive's representation, like LLVM's, is based on directed graphs of instructions in Static Single Assignment (SSA) form, the ordering of non-dependent instructions is irrelevant.

Alive implements a subset of LLVM's integer and pointer instructions. It also has limited support for branches: to avoid loops, they are not allowed to jump backwards. Alive supports LLVM's `nsw`, `nuw`, and `exact` instruction attributes that weaken the behavior of integer instructions by adding undefined behaviors.

Scoping. The source and target templates must have a common *root variable* that is the root of the respective graphs. The remaining variables are either inputs to the transformation or else temporary variables produced by instructions in the source or target template. Inputs are visible throughout the source and target templates. Temporaries defined in the source template are in scope for the precondition, the target, and the remaining part of the source from the point of definition. Temporaries declared in the target are in scope for the remainder of the target. To help catch errors, every temporary in the source template must be used in a later source instruction or be overwritten in the target, and all target instructions must be used in a later target instruction or overwrite a source instruction.

Constant expressions. To allow algebraic simplifications and constant folding, Alive includes a language for constant expressions. A constant expression may be a literal, an abstract constant (e.g., C in the example on the previous page), or a unary or binary operator applied to 1 or 2 constant

expressions. The operators include signed and unsigned arithmetic operators and bitwise logical operators. Alive also supports functions on constant expressions. Built-in functions include type conversions and mathematical and bitvector utilities (e.g., `abs()`, `umax()`, `width()`).

2.2. Type system

Alive supports a subset of LLVM's types, such as integers and pointers. LLVM uses arbitrarily-sized integers, with a separate type for each width (e.g., `i8` or `i57`). Alive has limited support for LLVM's pointer and array types, and does not support structures or vectors. Recent efforts have subsequently extended Alive with support for floating-point types.^{14,16}

Unlike LLVM, Alive permits type annotations to be omitted and does not require values to have a unique type. This enables succinct specifications of optimizations in Alive, as many peephole optimizations are not type-specific. A set of possible types is inferred for each implicitly-typed value, and the correctness of an optimization is checked for each type assignment. Because LLVM has infinitely many integer types, we set an upper bound of 64 bits for implicitly typed integer values.

2.3. Built-In predicates

Some peephole optimizations use the results of data-flow analyses. Alive makes these results available using a collection of built-in predicates such as `isPowerOf2()`, `MaskedValueIsZero()`, and `WillNotOverflowSignedAdd()`. The analyses producing these results are trusted by Alive: verifying their correctness is not within Alive's scope. Predicates can be combined with the usual logical connectives. Figure 2 shows an example transformation that includes a built-in predicate in its precondition.

2.4. Undefined behaviors in LLVM

To aggressively optimize well-defined programs, LLVM has 3 distinct kinds of undefined behavior. Together, they enable many desirable optimizations, and LLVM aggressively exploits these opportunities.

Undefined behavior in LLVM resembles undefined behavior in C and C++: anything may happen to a program that executes it. The compiler may simply assume that undefined behavior does not occur; this assumption places a

Figure 2. An example illustrating many of Alive's features. $((B \vee V) \wedge C1) \vee (B \wedge C2)$ can be transformed to $(B \vee V) \wedge (C1 \vee C2)$ when $C1 \wedge C2 = 0$ and when the predicate `MaskedValueIsZero(V, ~C1)` is true, indicating that an LLVM dataflow analysis has concluded that $V \wedge \neg C1 = 0$. $\%B$ and $\%V$ are input variables. $C1$ and $C2$ are constants. $\%t0$, $\%t1$, and $\%t2$ are temporaries. This transformation is rooted at $\%R$.

```
Pre: C1 & C2 == 0 && MaskedValueIsZero(%V, ~C1)
%t0 = or %B, %V
%t1 = and %t0, C1
%t2 = and %B, C2
%R = or %t1, %t2
=>
%R = and %t0, (C1 | C2)
```

^a The latest version of Alive can be found at <https://github.com/nunoplopes/alive>.

corresponding obligation on the program developer (or on the compiler and language runtime, when a safe language is compiled to LLVM) to ensure that undefined operations are never executed. An instruction that executes undefined behavior can be replaced with an arbitrary sequence of instructions. When an instruction executes undefined behavior, all subsequent instructions can be considered undefined as well.

Table 1 shows when Alive's arithmetic instructions have defined behavior, following the LLVM IR specification. For example, the `udiv` instruction is defined only when the dividend is non-zero. With the exception of memory access instructions (discussed in the original paper¹⁰), instructions not listed in Table 1 are always defined.

The *undefined value* (`undef` in the IR) is a limited form of undefined behavior that mimics a free-floating hardware register that can return any value each time it is read. Semantically, `undef` stands for the set of all possible bit patterns for a particular type; the compiler is free to pick a convenient value for each use of `undef` to enable aggressive optimizations. For example, a 1-bit undefined value, sign-extended to 32 bits, produces a variable containing either all 0s or all 1s.

Poison values, which are distinct from undefined values, are used to indicate that a side-effect-free instruction has a condition that produces undefined behavior. When the poison value gets used by an instruction with side effects, the program exhibits true undefined behavior. Hence, poison values are deferred undefined behaviors: they are intended to support speculative execution of possibly-undefined operations. Poison values taint subsequent dependent instructions; unlike `undef`, poison values cannot be untainted by subsequent operations. The subtleties in the semantics of `undef` and poison values and its impact on either enabling or disabling optimizations are currently being explored.⁸

Shift instructions, `shl`, `ashr`, and `lshr`, produce a poison value when their second argument, the *shift amount*, is larger than or equal to the bit width of the operation.

Instruction attributes modify the behavior of some LLVM instructions. The `nsw` attribute ("no signed wrap") makes signed overflow undefined. For example, this Alive transformation, which is equivalent to the optimization of `(x+1) > x` to `1` in C and C++ where `x` is a signed integer, is valid:

```
%1 = add nsw %x, 1
%2 = icmp sgt %1, %x
=>
%2 = true
```

Table 1. The constraints for arithmetic instructions to be defined. $<_u$ is unsigned less-than. INT_MIN is the smallest signed integer value for a given bitwidth.

Instruction	Definedness constraint
<code>sdiv a, b</code>	$b \neq 0 \wedge (a \neq INT_MIN \vee b \neq -1)$
<code>udiv a, b</code>	$b \neq 0$
<code>srem a, b</code>	$b \neq 0 \wedge (a \neq INT_MIN \vee b \neq -1)$
<code>urem a, b</code>	$b \neq 0$

An analogous `nsw` attribute exists to rule out unsigned wrap. If an `add`, `subtract`, `multiply`, or `left shift` operation with an `nsw` or `nuw` attribute overflows, the result is poison. Additionally, LLVM's right shift and divide instructions have an `exact` attribute that requires an operation to not be lossy. Table 2 provides the constraints for the instructions to be poison-free. Developers writing Alive patterns can omit instruction attributes, in which case Alive infers where they can be safely placed.

3. VERIFYING OPTIMIZATIONS IN ALIVE

The Alive interpreter verifies a transformation by automatically encoding the source and target, their definedness conditions, and the overall correctness criteria into SMT queries. An Alive transformation is parametric over the set of all *feasible types*: the concrete types satisfying the constraints of LLVM's type system and not exceeding the default limit of 64 bits.

In the absence of undefined behavior in the source or target of an Alive transformation, we can check correctness using a straightforward equivalence check: for each valuation of the input variables, the value of any variable that is present in both the source and target must be identical. However, an equivalence check is not sufficient to prove correctness in the presence of any of the 3 kinds of undefined behavior described in Section 2.4. We use refinement to reason about optimizations in the presence of undefined behavior. The target of an Alive transformation *refines* the source template if all the behaviors of the target are included in the set of behaviors of the source. That is, a transformation may remove undefined behaviors but not add them.

When an optimization contains or may produce `undef` values, we need to ensure that the target never produces a value that the source does not produce. In other words, an `undef` in the source represents a set of values and the target can refine it to any particular value, but an `undef` in the

Table 2. The constraints for arithmetic instructions to be poison-free. $>>_u$ and $\div_u$ are the unsigned shift and division operations. B is the bitwidth of the operands. $SExt(a, n)$ sign-extends a by n bits; $ZExt(a, n)$ zero-extends a by n bits.

Instruction	Constraints for poison-free execution
<code>add nsw a, b</code>	$SExt(a, 1) + SExt(b, 1) = SExt(a + b, 1)$
<code>add nuw a, b</code>	$ZExt(a, 1) + ZExt(b, 1) = ZExt(a + b, 1)$
<code>sub nsw a, b</code>	$SExt(a, 1) - SExt(b, 1) = SExt(a - b, 1)$
<code>sub nuw a, b</code>	$ZExt(a, 1) - ZExt(b, 1) = ZExt(a - b, 1)$
<code>mul nsw a, b</code>	$SExt(a, B) \times SExt(b, B) = SExt(a \times b, B)$
<code>mul nuw a, b</code>	$ZExt(a, B) \times ZExt(b, B) = ZExt(a \times b, B)$
<code>sdiv exact a, b</code>	$(a \div b) \times b = a$
<code>udiv exact a, b</code>	$(a \div_u b) \times b = a$
<code>shl a, b</code>	$b <_u B$
<code>shl nsw a, b</code>	$b <_u B \wedge (a << b) >> b = a$
<code>shl nuw a, b</code>	$b <_u B \wedge (a << b) >>_u b = a$
<code>ashr a, b</code>	$b <_u B$
<code>ashr exact a, b</code>	$b <_u B \wedge (a >> b) << b = a$
<code>lshr a, b</code>	$b <_u B$
<code>lshr exact a, b</code>	$b <_u B \wedge (a >>_u b) << b = a$

target represents a set of values which must *all* be refinements of the source. Poison values are handled by ensuring that an instruction in the target template will not yield a poison value when the source instruction did not, for any specific choice of input values. In summary, we check correctness by checking (1) the target is defined when the source is defined, (2) the target is poison-free when the source is poison-free, and (3) the target produces a subset of the values produced by the source when the source is defined and poison-free.

To determine whether these conditions hold, we ask an SMT solver to find cases where they are violated. When found, these counter-examples are reported to the user, as shown in Figure 3. Conversely, if the SMT solver can show that no counter-example exists, then the conditions must hold and the optimization is valid.

3.1. Verification condition generation

Alive's Verification Condition Generator (VC Gen) encodes the values, instructions, and expressions in a transformation into SMT expressions using the theory of bitvectors. The correspondence between LLVM operations and bitvector logic is very close, which makes the encoding straightforward. For each instruction, the interpreter computes 3 SMT expressions: (1) an expression ι for the result of the instruction, (2) an expression δ indicating whether the instruction is defined, and (3) an expression ρ indicating whether the result is free of poison. The first has a type corresponding to the return type of the instruction. The others are Boolean predicates. All 3 may contain free variables, corresponding to the uninterpreted input variables and symbolic constants in the optimization.

Typically, an instruction's result is encoded by applying the corresponding bitvector operation to the encoding of its arguments. Its definedness and poison-free conditions are the conjunction of the definedness and poison-free conditions, respectively, of its arguments along with any specific requirements for that instruction.

For example, consider the instruction `udiv exact %a, %b`, which is encoded as follows,

Figure 3. Alive's counterexample for the incorrect transformation reported as LLVM PR21245.

```
Pre: C2 % (1 << C1) == 0
%s = shl nsw %X, C1
%r = sdiv %s, C2
=>
%r = sdiv %X, C2 / (1 << C1)

ERROR: Mismatch in values of i4 %r

Example:
%X i4 = 0xF (15, -1)
C1 i4 = 0x3 (3)
C2 i4 = 0x8 (8, -8)
%s i4 = 0x8 (8, -8)
Source value: 0x1 (1)
Target value: 0xF (15, -1)
```

$$\begin{aligned}\iota &= \iota_a \div_u \iota_b \\ \delta &= \delta_a \wedge \delta_b \wedge \iota_b \neq 0 \\ \rho &= \rho_a \wedge \rho_b \wedge (\iota_a \div_u \iota_b) \times \iota_b = \iota_a,\end{aligned}$$

where $\div_u$ is unsigned bitvector division. The unsigned division requires the second argument to be non-zero, and the exact attribute requires %a to be divisible by %b.

Encoding undefvalues. Undefvalues represent sets of possible values. The VC Gen encodes them as fresh SMT variables, which are collected in a set $\mathcal{U}$. In particular, each reference to an undef value, direct or indirect, must receive a fresh SMT variable. The sets collected for the source and target will then be appropriately quantified over the correctness conditions.

Encoding preconditions. Alive's precondition sublanguage provides comparison operators and a set of named predicates, along with conjunction, disjunction, and negation. Aside from the predicates, these have a straightforward encoding in SMT.

The encoding of named predicates depends on whether the underlying analysis is precise or is an over- or under-approximation. For example, the predicate `isPower2` is implemented in LLVM with a must-analysis, that is, when `isPower2 (%a)` is true, we know for sure that %a is a power of 2; when it is false, no inference can be made. The VC Gen encodes the result of `isPower2 (%a)` using a fresh Boolean variable p , and a side constraint $p \Rightarrow a \neq 0 \wedge a \& (a - 1) = 0$.

The encoding of may-analyses is similar. The VC Gen creates a fresh variable p to represent the result of the analysis and a side constraint of the form $s \Rightarrow p$ where s is an expression summarizing the may-analysis based on the inputs. For example, a simplified encoding of `mayAlias (%a, %b)` is $a = b \Rightarrow p$.

Most analyses in LLVM are precise when their inputs are compile-time constants. Therefore, we encode the result of these analyses precisely when we detect such cases (done statically by the VC Gen).

3.2. Correctness criteria

Let ϕ be the encoding of the precondition, let ι_s and ι_t be the values computed by the source and target, respectively, and similarly let $\delta_s, \delta_t, \rho_s$, and ρ_t be the definedness and poison-free conditions.

Let $\mathcal{I}$ be the set of input variables, $\mathcal{P}$ be the Boolean variables used to encode analyses, and $\mathcal{U}_s$ and $\mathcal{U}_t$ be the sets of variables used to encode undef values in the source and target, respectively.

An Alive optimization is correct if and only if the following conditions hold for every feasible type assignment:

1. $\forall_{\mathcal{I}, \mathcal{P}, \mathcal{U}_t} \exists_{\mathcal{U}_s} : \phi \wedge \delta_s \Rightarrow \delta_t$
2. $\forall_{\mathcal{I}, \mathcal{P}, \mathcal{U}_t} \exists_{\mathcal{U}_s} : \phi \wedge \delta_s \wedge \rho_s \Rightarrow \rho_t$
3. $\forall_{\mathcal{I}, \mathcal{P}, \mathcal{U}_t} \exists_{\mathcal{U}_s} : \phi \wedge \delta_s \wedge \rho_s \Rightarrow \iota_s = \iota_t$

The first condition requires the target to be defined whenever the source is defined. The second condition requires the target to be poison-free whenever the source is defined and poison-free. The third condition requires the source and target to compute the same result when the source is defined and poison-free. The constraints are only required to hold if the

precondition is satisfied, because the optimization will not be applied otherwise.

Note that the variables used to represent undef values for the source and target are existentially and universally quantified, respectively. When an undef term occurs in the target, the target must refine the source for all possible values the undef term might take. In contrast, an undef term in the source may be instantiated with any value which makes the optimization a refinement. The order of the quantifiers permit undef values in the source to have different instantiations, depending on the instantiation of the undef values in the target.

We now state the correctness criteria for an Alive transformation:

THEOREM 1 (SOUNDNESS). *If conditions 1–3 hold for every instruction in an Alive transformation (without memory operations) and for any valid type assignment, then the transformation is correct.*

3.3. Illustration of correctness checking

We illustrate the verification condition generation and correctness conditions with 2 examples.

```
Pre: C1 != 0 && C2 %u C1 == 0
%m = mul nuw %a, C1
%r = udiv %m, C2
=>
%r = udiv %a, C2 /u C1
```

This is encoded using the following definitions for $\%r$:

$$\begin{aligned}\phi &\equiv c_1 \neq 0 \wedge c_2 \bmod_u c_1 = 0 \\ \delta_s &\equiv c_2 \neq 0 \\ \delta_T &\equiv c_2 \div_u c_1 \neq 0 \\ \rho_s &\equiv ZExt(a, B) \times ZExt(c_1, B) = ZExt(a \times c_1, B) \\ \rho_T &\equiv \top \\ \iota_s &= (a \times c_1) \div_u c_2 \\ \iota_T &= a \div_u (c_2 \div_u c_1)\end{aligned}$$

Note that ρ_s has propagated the poison-free condition for $\%m$, and that the target is always poison free. The sets $\mathcal{U}_s, \mathcal{U}_T$, and $\mathcal{P}$ are empty, so the correctness conditions are:

$$\begin{aligned}\forall_{a, c_1, c_2} : \phi \wedge \delta_s &\Rightarrow \delta_T \\ \forall_{a, c_1, c_2} : \phi \wedge \delta_s \wedge \rho_s &\Rightarrow \rho_T \\ \forall_{a, c_1, c_2} : \phi \wedge \delta_s \wedge \rho_s &\Rightarrow \iota_s = \iota_T.\end{aligned}$$

The VC Gen tests these conditions by querying an SMT solver for counterexamples, using the negation of the conditions. These queries are

1. $\exists_{a, c_1, c_2} : \phi \wedge \delta_s \wedge \neg \delta_T$
2. $\exists_{a, c_1, c_2} : \phi \wedge \delta_s \wedge \rho_s \wedge \neg \rho_T$
3. $\exists_{a, c_1, c_2} : \phi \wedge \delta_s \wedge \rho_s \wedge \iota_s \neq \iota_T$.

Since an SMT solver can prove that these formulas are unsatisfiable, then no counter-examples exist and therefore the optimization is correct for this type assignment.

Example with undef. The following simple optimization illustrates the nested quantifiers associated with undef:

```
%r = mul %x, undef
=>
%r = undef
```

There is no precondition, and the source and target are always defined and poison-free, so we need only consider the third correctness condition:

$$\forall_{x, u_2} \exists_{u_1} : x \times u_1 = u_2,$$

where u_1 and u_2 encode the undef values in the source and target, respectively. The corresponding SMT query is $\exists_{x, u_2} \forall_{u_1} : x \times u_1 \neq u_2$, which is satisfiable for $x=2, u_2=1$ (because multiplying by 2 always yields an even number). Thus, this optimization is incorrect.

3.4. Generating counterexamples

When Alive fails to prove the correctness of a transformation, it prints a counterexample showing values for inputs and constants, as well as for each of the preceding intermediate operations. We bias the SMT solver to produce counterexamples with bitwidths such as 4 or 8 bits. It is obvious that large-bitwidth examples are difficult to understand; we also noticed that, perhaps counter-intuitively, examples involving 1- or 2-bit variables are also not easy to understand, perhaps because almost every value is a corner case. Figure 3 shows an example.

4. GENERATING C++ FROM ALIVE

Optimizations specified in Alive can be directly translated into an implementation using the same instruction

Figure 4. An Alive transformation and its corresponding generated code. The C++ transformation is conditional on 2 match calls, one for each instruction in the source template, and also on the precondition. The target template has a single instruction and creates a new compile-time constant; both of these are directly reflected in the body of the C++ transformation.

Alive transformation:

```
Pre: isSignBit(C1)
%b = xor %a, C1
%d = add %b, C2
=>
%d = add %a, C1 ^ C2
```

Generated C++:

```
Value *a, *b;
ConstantInt *C1, *C2, *C3;

if (match(I, m_Add(m_Value(b), m_ConstantInt(C2))) &&
    match(b, m_Xor(m_Value(a), m_ConstantInt(C1))) &&
    C1->getValue().isSignBit()) {

    C3 = ConstantInt::get(I->getType(),
        C1->getValue() ^ C2->getValue());

    I->replaceAllUsesWith(
        BinaryOperator::CreateAdd(a, C3, "", I));
}
```

pattern-matching library that InstCombine uses. The implementation checks whether a code fragment matches the pattern of the source template and whether the precondition holds. If so, it creates the instructions in the target template, replacing variables with their corresponding values from the code fragment. Figure 4 shows an Alive transformation and its corresponding C++ implementation.

4.1. Translating a source template

The code generator uses LLVM's pattern-matching library to create a conditional which tests whether a code fragment matches the source template. For example, `match(I, m_Add(m_Value(b), m_ConstantInt(C2)))` returns true if the LLVM instruction `I` adds a value to a constant, and sets the variables `b` and `C2` to point to its arguments. Matching begins with the root instruction in the source template and recursively matches operands until all non-inputs have been bound.

4.2. Translating a target template

A new instruction is created for each instruction that is in the target template but not the source. The root instruction from the source is replaced by its counterpart in the target.

4.3. Type unification

The LLVM constructors for constant literal values and conversion instructions require explicit types. In general, this information will depend on types in the source. As Alive transformations are parametric over types, and Alive provides support for explicit and named types, such information is not readily available. The Alive code generator uses a unification-based type inference algorithm to identify appropriate types for the operands and introduces additional clauses in the `if` condition to ensure the operands have the appropriate type before invoking the transformation. This type system ensures that the generated code does not produce ill-typed LLVM code.

The unification proceeds in 3 phases. First, the types of the operands in the source are unified according to the constraints in the source (e.g., the operands of a binary operator must have the same type) based on the assumption that source is a well-formed LLVM program. Second, the types of the operands in the target are similarly unified according to constraints of the target. Third, when the operands of a particular instruction in the target do not belong to the same class, then an explicit check requiring that the types are equal is added to the `if` condition in the C++ code generated. The explicit check is necessary as the target has type constraints that cannot be determined by the source alone.

5. IMPLEMENTATION

We implemented Alive in Python and used the Z3 SMT solver⁴ to discharge both typing and refinement constraints. Alive is about 5,200 lines of open-source code.

The number of possible type assignments for a transformation is usually infinite. To ensure termination, Alive considers integer types up to 64 bits.

⁴ The version of Alive corresponding to this paper can be found at <https://github.com/nunoplopes/alive/tree/pldi15>.

Refinement constraints are either over the BV or QF_BV (quantified/quantifier-free bitvector) theories. The constraints in Section 3.2 are negated before querying the SMT solver, effectively removing one quantifier alternation. Therefore, for transformations without undefined values in the source template, we obtain quantifier-free formulas, and formulas with a single quantifier otherwise.

6. EVALUATION^b

We translated hundreds of peephole optimizations from LLVM into Alive. We verified them, and we translated the Alive optimizations into C++ that we linked into LLVM and then used the resulting optimizer to build LLVM's test suite and the SPEC INT 2000 and 2006 benchmarks. The Alive-generated C++ code's compilation time and the performance of the resultant code compiled with it is similar to LLVM's unverified InstCombine pass.¹⁰

6.1. Translating and verifying InstCombine

LLVM's InstCombine pass rewrites expression trees to reduce their cost, but does not change the control-flow graph. During the initial testing of our prototype, we translated 334 InstCombine transformations to Alive. Of these, 8 could not be proved correct. We reported these erroneous transformations to the LLVM developers, who confirmed and fixed them. We re-translated the fixed optimizations to Alive and proved them correct.

Subsequent efforts have used Alive to validate end-to-end transformations and extended the Alive language. These have lead to the discovery of at least fifteen additional bugs.

The buggiest InstCombine file that we found was `MulDivRem`, which implements optimizations that have multiply, divide, and remainder instructions as the root of expression trees. Out of the 44 translated optimizations, we found that 6 of them (14%) were incorrect.

The most common kind of bug in InstCombine was the introduction of undefined behavior, where an optimization replaces an expression with one that is defined for a smaller range of inputs than was the original expression. There were 4 bugs in this category. We also found 2 bugs where the value of an expression was incorrect for some inputs, and 2 bugs where a transformation would generate a poison value for inputs that the original expression did not. Figure 5 provides the Alive code and the bug report numbers for a sample of the bugs that we discovered during our translation of LLVM InstCombine optimizations into Alive.

Alive usually takes a few seconds to verify the correctness of a transformation, during which time it may issue hundreds or thousands of incremental solver calls. Unfortunately, for some transformations involving multiplication and division instructions, Alive can take several hours or longer to verify the larger bitwidths. This indicates that further improvements are needed in SMT solvers to efficiently handle such formulas. In the meantime, we work around slow verifications by limiting the bitwidths of operands.

6.2. Preventing new bugs

Several LLVM developers use Alive to avoid introducing wrong-code bugs. Also, we have been monitoring proposed LLVM

patches and trying to catch incorrect transformations before they are committed to the tree. For example, in August 2014 a developer submitted a patch that improved the performance of one of the SPEC CPU 2000 benchmarks by 3.8%—this is obviously an interesting addition to a compiler. Using Alive, we discovered that the developer’s initial and second patches were wrong, and we proved that the third one was correct. This third and final patch retained the performance improvement without compromising the correctness of LLVM. Figure 6 shows the initially proposed optimization, a counter-example demonstrating its invalidity, and the final precondition for the valid optimization. A recent work has proposed a learning technique for automatically inferring preconditions, which is useful to developers debugging an incorrect optimization.¹³

7. RELATED WORK

Prior research on improving compiler correctness can be broadly classified into compiler testing tools, formal reasoning frameworks for compilers, and Domain Specific Languages (DSLs). DSLs for compiler optimizations are the most closely related work to Alive. Among them, Alive is perhaps most similar to high-level rewrite patterns.^{6,11} Alive differs in its extensive treatment of undefined behavior, which is heavily exploited by LLVM and other aggressive modern compilers, and its ability to generate code that is similar to LLVM’s InstCombine pass.

Peephole optimization patterns for a particular Instruction Set Architecture (ISA) can be generated from an ISA specification.³ In contrast to compiler optimizations, optimized code sequences can be synthesized either with peephole pattern generation or through superoptimization.^{1,5,12,19}

Optgen² automatically generates peephole optimizations. Like Alive, Optgen operates at the IR level and uses SMT solvers to verify the proposed optimizations. While Alive focuses on verifying developer-created optimizations, Optgen generates all possible optimizations up to a specified cost and can generate a test suite to check optimizations not implemented in a given compiler. In contrast to Alive, Optgen handles only integer operations and does not handle memory operations, poison values, support any operation producing undefined behavior, or abstraction over bitwidths/types.

Random testing tools^{7,15,20} have discovered numerous bugs in LLVM optimizations both for sequential programs and concurrent programs. These tools are not complete, as was shown by the bugs we found in optimizations that had previously been fuzzed.

An alternative approach to compiler correctness is translation validation^{17,18} where, for each compilation, it is proved that the optimized code refines the unoptimized code. Translation validation suffers from the drawback of requiring proof machinery to execute during every compilation. Alive aims for once-and-for-all proof of correctness of a limited slice of the compiler.

Figure 5. A sample of incorrect InstCombine transformations discovered during the development of Alive.

<pre>Name: PR20186 %a = sdiv %X, C %r = sub 0, %a => %r = sdiv %X, -C</pre>	<pre>Name: PR20189 %B = sub 0, %A %C = sub nsw %x, %B => %C = add nsw %x, %A</pre>	<pre>Name: PR21242 Pre: isPowerOf2(C1) %r = mul nsw %x, C1 => %r = shl nsw %x, log2(C1)</pre>	<pre>Name: PR21243 Pre: !WillNotOverflowSignedMul(C1, C2) %Op0 = sdiv %X, C1 %r = sdiv %Op0, C2 => %r = 0</pre>
<pre>Name: PR21245 Pre: C2 % (1<<C1) == 0 %s = shl nsw %X, C1 %r = sdiv %s, C2 => %r = sdiv %X, C2/(1<<C1)</pre>	<pre>Name: PR21255 %Op0 = lshr %X, C1 %r = udiv %Op0, C2 => %r = udiv %X, C2 << C1</pre>	<pre>Name: PR21256 %Op1 = sub 0, %X %r = srem %Op0, %Op1 => %r = srem %Op0, %X</pre>	<pre>Name: PR21274 Pre: isPowerOf2(%Power) && hasOneUse(%Y) %s = shl %Power, %A %Y = lshr %s, %B %r = udiv %X, %Y => %sub = sub %A, %B %Y = shl %Power, %sub %r = udiv %X, %Y</pre>

Figure 6. (a) A peephole optimization proposed by the developer. (b) A counterexample found by Alive. (c) A precondition that makes the optimization valid.

(a)	(b)	(c)
<pre>Pre: isPowerOf2(C1 ^ C2) %x = add %A, C1 %i = icmp ult %x, C3 %y = add %A, C2 %j = icmp ult %y, C3 %r = or %i, %j => %and = and %A, ~(C1 ^ C2) %lhs = add %and, umax(C1, C2) %r = icmp ult %lhs, C3</pre>	<pre>ERROR: Mismatch in values of i1 %r Example: %A i4 = 0x5 (5) C1 i4 = 0x3 (3) C3 i4 = 0x7 (7) C2 i4 = 0x1 (1) %x i4 = 0x8 (8, -8) %i i1 = 0x0 (0) %y i4 = 0x6 (6) %j i1 = 0x1 (1, -1) %and i4 = 0x5 (5) %lhs i4 = 0x8 (8, -8) Source value: 0x1 (1, -1) Target value: 0x0 (0)</pre>	<pre>Pre: C1 u> C3 && C2 u> C3 && isPowerOf2(C1 ^ C2) && isPowerOf2(-C2 ^ -C1) && -C2 ^ -C1 == (C3-C2) ^ (C3-C1) && abs(C1-C2) u> C3</pre>

The CompCert⁹ compiler for C is an end-to-end verified compiler developed with the interactive proof assistant Coq. Vellvm²¹ reuses the memory model from the CompCert development and formalizes the semantics and SSA properties of the LLVM IR to reason about optimizations. Alive's treatment of `undef` values mirrors the treatment in Vellvm. In contrast to Vellvm, Alive handles poison values and automates reasoning with an SMT solver.

8. CONCLUSION

We have shown that an important class of optimizations in LLVM—peephole optimizations—can be formalized in Alive, a new language that specifies optimizations more concisely than C++ code, while also supporting automated proofs of correctness. We designed Alive to resemble LLVM's textual format while also supporting abstraction over types and constant values. After an Alive transformation has been proved correct, it can be automatically translated into C++ that can be included in an optimization pass. Our first goal was to create a tool that is useful for LLVM developers. We believe this goal has been accomplished, as LLVM developers are actively using it. Second, we would like to see a large part of InstCombine replaced with code generated by Alive; we are still working towards that goal.

Acknowledgments

The authors thank the LLVM developers for their continued support for the development of Alive, for discussions regarding LLVM IR's semantics, and for adopting Alive. A special thanks to David Majnemer for confirming and fixing the bugs we reported. Eric Eide and Raimondas Sasnauskas provided valuable feedback on this work. This paper is based upon work supported in part by NSF CAREER Award CCF-1453086, NSF Award CNS-1218022, and a Google Faculty Award.

References

- Bansal, S., Aiken, A. Automatic generation of peephole superoptimizers. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (2006), 394–403.
- Buchwald, S. Optgen: A generator for local optimizations. In *Proceedings of the 24th International Conference on Compiler Construction (CC)* (2015).
- Davidson, J.W., Fraser, C.W. Automatic generation of peephole optimizations. In *Proceedings of the 1984 SIGPLAN Symposium on Compiler Construction* (1984).
- De Moura, L., Bjørner, N. Z3: An efficient SMT solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (2008), 337–340.
- Joshi, R., Nelson, G., Zhou, Y., Denali: A practical algorithm for generating optimal code. *ACM Trans. Program. Lang. Syst.* 28, 6 (Nov. 2006), 967–989.
- Kundu, S., Tatlock, Z., Lerner, S. Proving optimizations correct using parameterized program equivalence. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation* (2009).
- Le, V., Afshari, M., Su, Z. Compiler validation via equivalence modulo inputs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation* (2014).
- Lee, J., Kim, Y., Song, Y., Hur, C.-K., Das, S., Majnemer, D., Regehr, J., Lopes, N.P. Taming undefined behavior in LLVM. In *Proceedings of the 38th annual ACM SIGPLAN conference on Programming Language Design and Implementation* (2017).
- Leroy, X. Formal verification of a realistic compiler. *Commun. of the ACM* 52, 7 (2009), 107–115.
- Lopes, N.P., Menendez, D., Nagarakatte, S., Regehr, J. Provably correct peephole optimizations with Alive. In *Proceedings of the 36th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)* (2015).
- Lopes, N.P., Monteiro, J. Automatic equivalence checking of programs with uninterpreted functions and integer arithmetic. *Int. J. Softw. Tools Technol. Transf.* 18, 4 (2016), 359–374.
- Massalin, H. Superoptimizer: A look at the smallest program. In *Proceedings of the 2nd International Conference on Architectural Support*

- for Programming Languages and Operating Systems (ASPLOS) (1987).
- Menendez, D., Nagarakatte, S. Alive-infer: Data-driven precondition inference for peephole optimizations in LLVM. In *Proceedings of the 38th annual ACM SIGPLAN conference on Programming Language Design and Implementation* (2017).
- Menendez, D., Nagarakatte, S., Gupta, A. Alive-FP: Automated verification of floating point based peephole optimizations in LLVM. In *Proceedings of the 23rd International Symposium on Static Analysis* (2016).
- Morisset, R., Pawan, P., Nardelli, F.Z. Compiler testing via a theory of sound optimisations in the C11/C++11 memory model. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation* (2013).
- Nötzli, A., Brown, F. LifeJacket: Verifying precise floating-point optimizations in LLVM. In *Proceedings of the 5th ACM SIGPLAN International Workshop on State Of the Art in Program Analysis* (2016).
- Phueli, A., Siegel, M., Singerman, E. Translation validation. In *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems* (1998), 151–166.
- Samet, H. Proving the correctness of heuristically optimized code. In *Communications of the ACM* (1978).
- Schkufza, E., Sharma, R., Aiken, A. Stochastic superoptimization. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (2013).
- Yang, X., Chen, Y., Eide, E., Regehr, J. Finding and understanding bugs in C compilers. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (2011).
- Zhao, J., Nagarakatte, S., Martin, M.M., Zdanciewicz, S. Formalizing the LLVM intermediate representation for verified program transformations. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (2012).

Nuno P. Lopes (nlopes@microsoft.com),
Microsoft Research, UK.
David Menendez and Santosh
Nagarakatte ([davemm, santosh.

nagarakatte]@cs.rutgers.edu), Rutgers
University, USA.
John Regehr (regehr@cs.utah.edu),
University of Utah, USA.

© 2018 ACM 0001-0782/18/2 \$15.00



ECSEE

European Conference Software Engineering Education

14 and 15
June 2018

Seeon Monastery
Germany

Full Paper
Submission Deadline:
16 March 2018

www.ecsee.eu

Proceedings will be published in:



Published by ACM

Technical Perspective

Designing Algorithms and the Fairness Criteria They Should Satisfy

By Vincent Conitzer

ALGORITHMS ARE INCREASINGLY used to determine allocations of scarce, high-value resources. For example, spectrum auctions, which are used by governments to allocate radio spectrum, require algorithms to determine which combinations of bids can and should be accepted. Kidney exchanges allow patients that require a kidney transplant and have a willing but medically incompatible donor to trade their donors, and some of these exchanges now use algorithms to determine who matches with whom. These are very different application domains—for one, in the former, transfers of money play an essential role, but in the latter, they are illegal. Other applications have yet different features, so each application comes with its own requirements.

In spite of the lack of a single, universal solution, there are several clear benefits of allocating resources algorithmically. In both of the given examples, there is a combinatorial explosion in the space of possible alternatives, and computers are much better able to search through these spaces. If the algorithm is clearly specified beforehand, this can also improve trust in the process as a whole. On the other hand, the process of designing the algorithm often brings into sharp focus that the objective is not clear. Should a government running a spectrum auction focus on the spectrum being allocated efficiently, or on bringing in revenue? In a kidney exchange, should we simply maximize the number of transplants, or give some priority to certain patients, for example, ones who will be difficult to match in the future due to blood type?

Without answers to these questions, it is difficult to design the algorithm; even if we avoid explicitly answering them, any choice of algorithm implicitly corresponds to a decision about how much to priori-

If there is one objective that is often both essential and difficult to make precise, it is that of fairness.

tize each aspect. On top of that, different objectives often require algorithms that are quite different in nature, even in the same application domain—and some objectives will not allow a sufficiently efficient algorithm. As a result, having a clean division of labor between computer scientists who design the algorithms, and others (policymakers, economists, doctors, ethicists) who determine the objective(s) to optimize is generally not feasible. They need to talk with each other.

If there is one objective that is often both essential and difficult to make precise, it is that of fairness. Optimizing some straightforward criterion often results in outcomes that people intuitively perceive to be unfair. To make matters worse, it is often difficult for them to put their finger on precisely what makes these outcomes unfair. They may be able to verbalize it to some extent, but it will generally fall short of a precise mathematical criterion.

The paper that follows focuses on the specific problem of rent division, where we have n people renting an apartment together that costs B to rent. There are n bedrooms to assign among them, and the rooms are not all the same, so it may make sense to

split the rent unequally. However, the renters do not agree on how much each room is worth. Given how much each person values each room, how should the rent be split? This is the problem the authors set out to solve. They are motivated in part by the website Spliddit, which was developed by some of the authors to make tools for various kinds of fair division problems, including rent division, broadly available. While the rent division problem is important in its own right, in my mind the bigger contribution of the paper is as a great case study in how to address the types of problems discussed earlier. What does it mean for a rent division to be fair, and is there an efficient algorithm for computing such rent divisions?

What makes the paper stand out is the variety of techniques applied to arrive at a solution. The authors are guided by theory, including the Second Welfare Theorem from mathematical economics. But they also do a user study, with Spliddit users as the subjects. And, of course, they design an efficient algorithm for the problem, which is now deployed on Spliddit.

Algorithms make increasingly many decisions about allocations of resources to people, as well as other decisions affecting people's lives—for example, machine learning classifiers determining whether someone is released on bail. The type of multidisciplinary approach in the following paper—combining techniques from economic theory, the behavioral sciences, and computer science—is essential for steering these developments in the most beneficial direction. 

Vincent Conitzer (conitzer@cs.duke.edu) is the Kimberly J. Jenkins University Professor of New Technologies at Duke University, Durham, NC, USA.

Copyright held by author.

Which Is the Fairest (Rent Division) of Them All?

By Kobi Gal, Ariel D. Procaccia, Moshe Mash, and Yair Zick

“Mirror mirror on the wall, who is the fairest of them all?”

The Evil Queen

Abstract

What is a *fair* way to assign rooms to several housemates, and divide the rent between them? This is not just a theoretical question: many people have used the *Spliddit* website to obtain *envy-free* solutions to rent division instances. But envy freeness, in and of itself, is insufficient to guarantee outcomes that people view as intuitive and acceptable. We therefore focus on solutions that optimize a criterion of social justice, subject to the envy freeness constraint, in order to pinpoint the “fairest” solutions. We develop a general algorithmic framework that enables the computation of such solutions in polynomial time. We then study the relations between natural optimization objectives, and identify the *maximin* solution, which maximizes the minimum utility subject to envy freeness, as the most attractive. We demonstrate, using experiments on real data from *Spliddit*, that the *maximin* solution gives rise to significant gains in terms of our optimization objectives. Finally, a user study with *Spliddit* users as subjects demonstrates that people find the *maximin* solution to be significantly fairer than arbitrary *envy-free* solutions; this user study is unprecedented in that it asks people about their real-world rent division instances. Based on these results, the *maximin* solution has been deployed on *Spliddit* since April 2015.

1. INTRODUCTION

Many a reader may have personally experienced the *rent division problem*: several housemates move in together, and need to decide who gets which room, and at what price. The problem becomes interesting—and, more often than not, a source of frustration—when the rooms differ in quality. The challenge is then to achieve “rental harmony”¹⁸ by assigning the rooms and dividing the rent *fairly*.

In more detail, suppose each player i has value v_{ij} for room j , such that each player’s values for the rooms sum up to the total rent (see Figure 1a). The (quasilinear) utility of player i for getting room j at price p_j is $v_{ij} - p_j$ (see Figure 1b). A solution (i.e. an assignment of the rooms and division of the rent) is *envy free*⁸ if the utility of each player for getting his room at its price is at least as high as getting any other room at the price of that room (see Figure 1c). More generally, one can think of this problem as allocating indivisible goods and splitting a sum of money—but we adopt the rent division terminology, which grounds the problem and justifies our assumptions.

Envy freeness is undoubtedly a compelling fairness notion. But what makes it truly powerful in the context of rent division is that an *envy-free* solution to a rent division

problem always exists.¹⁹ Even better, such a solution can be computed in polynomial time.³

However, envy freeness in and of itself is insufficient to guarantee satisfactory solutions. For example, consider an apartment with three rooms and total rent of \$3000. Each player i has value \$3000 for room i , and value \$0 for the two other rooms. Furthermore, consider the solution that assigns room 1 to player 1 at \$3000, and, for $i \in \{2, 3\}$, gives room i to player i for free. This solution is *envy free*: players 2 and 3 are obviously overjoyed, while player 1 is indifferent between the three rooms. However, from an interpersonal perspective, this solution is not fair at all, as the distribution of prices between players is unequal. An intuitive alternative solution here would be to keep the same assignment of rooms, but equally split the rent between the different rooms—\$1000 per room—thereby equalizing the utilities of the players.

The challenge, therefore, is to choose among many possible *envy-free* solutions. And, arguably, the most natural way to do this is to optimize a function of the utilities that meets desirable social criteria, subject to the *envy freeness* constraint.² In particular, if we were to maximize the minimum utility of any player subject to *envy freeness*, or if we were to minimize the maximum difference in utilities subject to *envy freeness*, we would obtain the aforementioned solution in the example. This focus on optimization in rent division motivates us to

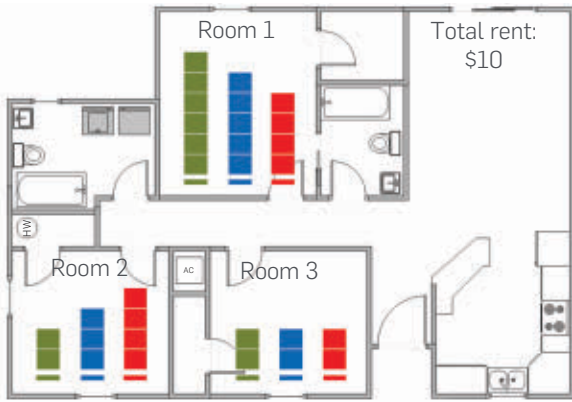
...design polynomial-time algorithms for optimization under the *envy freeness* constraint; understand the relationship between natural optimization objectives; and measure the benefits of optimization in rent division.

1.1. Real-world connections and implications: The *Spliddit* service

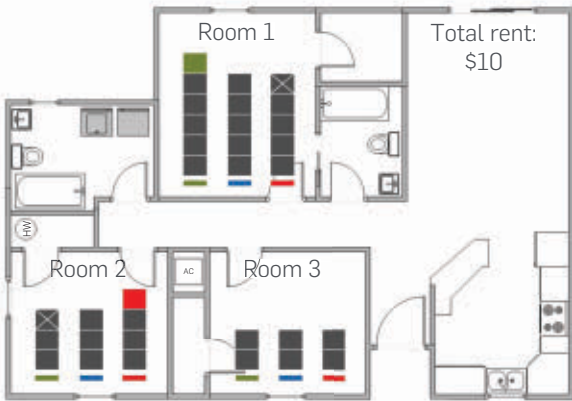
The above challenges are especially pertinent when put in the context of *Spliddit* (www.spliddit.org), a not-for-profit fair division website.⁹ *Spliddit* offers “provably fair solutions” for the division of credit, indivisible goods, chores, fare—and, of course, rent. Since its launch in November 2014, *Spliddit* has attracted more than 100,000 users, who, in particular, have created 27,344 rent division instances (as of July 6, 2017).

Until April 2015, *Spliddit*’s rent division application relied on the algorithm of Abdulkadiroğlu et al.,¹ which elicits the values of the players for the rooms, and computes an *envy-free* solution assuming quasi-linear utilities. While many users were satisfied with the results (based on their

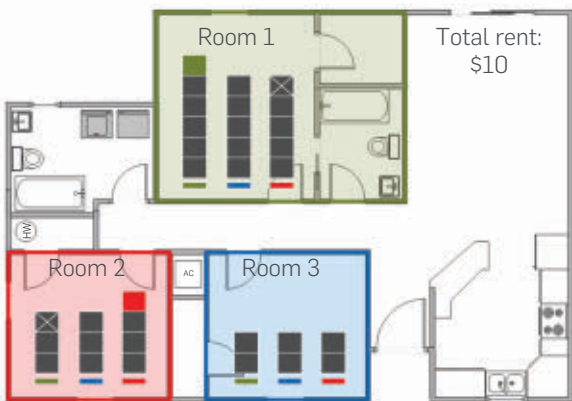
The original version of this paper was published in the *Proceedings of the 17th ACM Conference on Economics and Computation* (EC ’16).

Figure 1. Envy-free solutions, illustrated.

(a) Values, shown by colored boxes. For example, the values of the green, blue, and red players for Room 1 are 6, 5, and 4, respectively. Note that the values of each player for the three rooms add up to the total rent of 10.



(b) Prices, shown by gray boxes, and utilities. For example, the price of Room 1 is 5, and the utility of the green player for Room 1 at its price is $6 - 5 = 1$.



(c) An envy-free solution. For example, the green player is not envious because he has utility 1 for his room (Room 1), utility -1 for Room 2 (indicated by an "X" in the top box), and utility 0 for Room 3.

reported evaluations), the algorithm does provide nonintuitive solutions in some cases. This prompted an investigation of alternative approaches, and ultimately led to the deployment of a new algorithm in April 2015, based entirely on the results presented in (the original version of) this paper.

It is important to point out that Spliddit not only motivates our research questions, but also helps answer them. Indeed, while Spliddit's primary goals are making fair division methods accessible to people, and outreach, a secondary goal is the collection of an unprecedented dataset for fair division research.⁹ This real-world dataset is exciting because, as noted by Herreiner and Puppe,¹² fair division is hard to study in the lab: researchers can tell subjects what their valuations are for different goods, but these values are not ecologically realistic, in that they do not represent subjects' actual preferences. To quote Herreiner and Puppe,¹² "the goods in the lab are not really distributed among participants, but serve as temporary substitutes for money." In contrast, Spliddit instances are ecologically valid, as they are posed by real people facing real division problems. Thus the Spliddit data enables studies at a realistic level and scale that was not possible before. Even better, we can ask Spliddit users to evaluate different solutions based on the actual instances they participated in. This is exactly what we do in this paper.

1.2. Our results

We start, in Section 3, by constructing a general yet simple algorithmic framework for optimization under the envy freeness constraint. Specifically, our algorithm maximizes the minimum of linear functions of the utilities, subject to envy freeness, in polynomial time. We do this by using the Second Welfare Theorem to argue that we can employ any welfare-maximizing assignment of players to rooms, and then solve a linear program to compute the optimal envy free prices.^a

Our main goal in Section 4 is to understand the relation between two solution concepts: the *maximin* solution,² which maximizes the minimum utility of any player subject to envy freeness; and the *equitable* solution, which minimizes *disparity*—the maximum difference in utilities—subject to envy freeness. (Our algorithm can compute either solution in polynomial time.) Our most significant result in this section is proving that the maximin solution is also equitable, but not every equitable solution is maximin.

Based on these results, we have implemented the polynomial-time algorithm of Section 3, with the maximin objective function.^b As noted above, it has been deployed on Spliddit since April 2015.

^a It is interesting to note that, even though the instances on Spliddit are small, computational tractability does play a key role, as there are many instances and computation incurs a cost (Spliddit uses Amazon Web Services to run all its algorithms).

^b To be completely precise, the algorithm deployed on Spliddit first tries to maximize the minimum utility, subject to envy freeness as well as an additional constraint: prices must be non-negative. If an envy-free solution with non-negative prices does not exist [4], it removes the non-negative price constraint (in which case a solution always exists). Most of our results go through even when prices are assumed to be non-negative. In any case, real-world instances where negative prices actually help are extremely rare, so throughout the paper prices are unconstrained.

The remainder of the paper focuses on demonstrating that the foregoing approach is indeed effective. Here our contribution is twofold. First, we show that real-world instances give rise to significant differences, according to both the maximin and equitability objectives, between the maximin solution (which optimizes both objectives simultaneously) and an arbitrary envy-free solution (which does not attempt to optimize either objective).

Second, we report results from a user study. We contacted Spliddit users, and asked them to compare two solutions: the maximin solution, and an arbitrary envy-free solution. Crucially, the two solutions were computed on each user's actual Spliddit instance (the values of other tenants were perturbed to preserve privacy). Subjects were asked to subjectively rate the solutions in terms of fairness to themselves, and fairness to others. The results show a significant advantage for the maximin solution in both questions, thereby demonstrating the added value of optimization and supporting the decision to use the maximin solution on Spliddit.

1.3. Related work

The idea of refining envy-free solutions has been explored in several papers,^{2,20,21,22} typically from an axiomatic viewpoint. We focus on the work of Alkan et al.,² who study the more general problem of allocating goods and dividing money. They start by proving the existence of envy-free solutions in this setting, but, like us, they ultimately employ criteria of justice in order to find the "best" envy-free solutions. They are especially interested in the maximin solution, which they call the *value-Rawlsian* solution; and the solution that maximizes the minimum amount of money allocated to any player, subject to envy freeness, which they call the *money-Rawlsian* solution. They show that the maximin solution is unique, as are a number of less attractive solutions (minimize the maximum utility, maximize the utility of one particular player). Finally, they show that these criteria imply solutions with a monotonicity property: if the amount of money is increased, the utility of all players is strictly higher (this property is moot in our setting). Alkan et al.² do not provide algorithmic results.

Aragones³ designs a polynomial-time algorithm for computing the money-Rawlsian solution of Alkan et al.² Her combinatorial algorithm does not extend to other criteria. In contrast, our LP-based framework is significantly more general, and, in particular, allows us to compute the maximin solution (which we view as the most attractive) in polynomial time. Our algorithmic approach is also much simpler. It is worth noting that Klijn¹⁴ gives a different polynomial-time algorithm for computing envy-free solutions, without guaranteeing any additional properties (other than being extreme points of a certain polytope).

There are (at least) three *marketlike* mechanisms for computing solutions for the rent division problem assuming quasi-linear utilities, by Brams and Kilgour,⁴ Haake et al.,¹¹ and Abdulkadiroğlu et al.¹ All three do not consider optimization criteria; in the case of the mechanism of Brams and Kilgour,⁴ the solution may not be envy free. As mentioned above, the mechanism of Abdulkadiroğlu et al.¹ was deployed on Spliddit until April 2015.

One fundamentally different approach to rent division that we would like to discuss in more detail is that of Su.¹⁸ He does not assume quasi-linear utilities; rather, his main assumption is that a player would always prefer getting a free room to getting another room at a positive price (the so-called *miserly tenants* assumption). Under this assumption, Su¹⁸ designs an algorithm that converges to an (approximately) envy-free solution, by iteratively querying players about their favorite room at given prices. While eschewing the quasi-linear utilities assumption is compelling, a (crucial, in our view) disadvantage of this approach is that preference elicitation is very cumbersome. Interestingly, Su's method was implemented by the New York Times.^c

Relatively few papers explore fair allocations among people in lab settings, and there is inconclusive evidence about the types of solution criteria that are favored by people. Dupuis-Roy and Gosselin⁷ report that fair division algorithms were rated less desirable than imperfect allocations that did not employ any fairness criterion, while Schneider and Krämer¹⁷ find that subjects preferred envy-free solutions to a divide-and-choose method that does not guarantee envy freeness. Herreiner and Puppe^{12,13} find that envy freeness was a dominant factor in the allocations favored by subjects, but that it was a secondary criterion to Pareto optimality or inequality minimizing allocations. Kohler¹⁵ proposes an equilibrium strategy for repeated negotiation that incorporates fairness and envy concerns. In all of these papers, the studies were conducted in a controlled lab setting in which subjects' valuations over goods were imposed on the subjects, or the goods to be allocated were chosen by the experimenters themselves.

2. THE MODEL

We are interested in rent division problems involving a set of players $[n] = \{1, \dots, n\}$, and a set of rooms $[n]$. Each player i has a non-negative value $v_{ij} \in \mathbb{R}^+$ for each room j . We assume without loss of generality that the total rent is 1, and also assume (with loss of generality) that for all $i \in [n]$, $\sum_{j=1}^n v_{ij} = 1$. We can therefore represent an instance of the rent division problem as a right stochastic (rows sum to 1) matrix $V \in \mathbb{M}_{n \times n}(\mathbb{R}^+)$.

An assignment of the rooms is a permutation $\sigma: [n] \rightarrow [n]$, where $\sigma(i)$ is the room assigned to player i . The division of rent is represented through a vector of (possibly negative) prices $\mathbf{p} \in \mathbb{R}^n$ such that $\sum_{i=1}^n p_i = 1$; p_j is the price of room j .

Given a solution $(\sigma, \mathbf{p})$ for a rent division problem V , the quasi-linear *utility* of player i is denoted $u_i(\sigma, \mathbf{p}) = v_{i\sigma(i)} - p_{\sigma(i)}$. A solution is *Envy Free (EF)* if the utility of each player for her room is at least as high as any other room. Formally, $(\sigma, \mathbf{p})$ is EF if and only if

$$\forall i, j \in [n], v_{i\sigma(i)} - p_{\sigma(i)} \geq v_{ij} - p_j. \quad (1)$$

3. COMPUTATION OF OPTIMAL ENVY-FREE SOLUTIONS

As noted above, it is possible to compute an envy-free solution to a given rent division problem in polynomial time.³ We are interested in choosing among envy-free allocations by optimizing an objective function, subject to the envy freeness constraint. Our goal in this section is to show that this

^c <http://goo.gl/Xp3omV>. This article also discusses the then under-construction Spliddit.

can be done in polynomial time, when the objective function is the minimum of linear functions of the utilities.

THEOREM 3.1. *Let $f_1, \dots, f_t: \mathbb{R}^n \rightarrow \mathbb{R}$ be linear functions, where t is polynomial in n . Given a rent division instance V , a solution $(\sigma, \mathbf{p})$ that maximizes the minimum of $f_q(u_1(\sigma, \mathbf{p}), \dots, u_n(\sigma, \mathbf{p}))$ over all $q \in [t]$ subject to envy freeness can be computed in polynomial time.*

Natural examples of objective functions of the form specified in the theorem are maximizing the minimum utility, and minimizing the maximum difference in utilities; we discuss these objectives in detail in Section 4. The former objective can be directly captured by setting $t = n$, and $f_i(u_1(\sigma, \mathbf{p}), \dots, u_n(\sigma, \mathbf{p})) = u_i(\sigma, \mathbf{p})$ for all $i \in [n]$. The latter criterion is also captured by setting $t = n^2$ and,

$$f_{ij}(u_1(\sigma, \mathbf{p}), \dots, u_n(\sigma, \mathbf{p})) = u_i(\sigma, \mathbf{p}) - u_j(\sigma, \mathbf{p}).$$

Indeed,

$$\min_{i,j \in [n]} f_{ij}(u_1(\sigma, \mathbf{p}), \dots, u_n(\sigma, \mathbf{p})) = -\max_{i,j \in [n]} \{u_i(\sigma, \mathbf{p}) - u_j(\sigma, \mathbf{p})\},$$

so maximizing the minimum of these linear functions is equivalent to minimizing the maximum difference in utilities.

Our polynomial-time algorithm relies on a connection between envy-free rent division and the concept of *Walrasian equilibrium*. To understand this connection, imagine a more general setting where a set of buyers $[n]$ are interested in purchasing bundles of goods G ; here, each buyer i has a valuation function $v_i: 2^G \rightarrow \mathbb{R}$, assigning a value $v_i(S)$ to every bundle of goods. A Walrasian equilibrium is an allocation $\mathbf{A} = (A_1, \dots, A_n)$ of the goods to buyers (where $A_i \subseteq G$ is the bundle given to buyer i), coupled with a price vector $\mathbf{p}$ that assigns a price to each good, such that each player receives the best bundle of goods that she can buy for the price $\mathbf{p}$; formally:

$$\forall i \in [n], S \subseteq G, v_i(A_i) - p(A_i) \geq v_i(S) - p(S). \quad (2)$$

We say that an allocation $\mathbf{A}$ is *welfare-maximizing* if it maximizes $\sum_{i=1}^n v_i(A_i)$. The following properties of Walrasian equilibria are well known; see, for example, the book of Mas-Colell et al.¹⁶ (Chapter 16).

THEOREM 3.2 (1ST WELFARE THEOREM). *If $(\mathbf{A}, \mathbf{p})$ is a Walrasian equilibrium, then $\mathbf{A}$ is a welfare-maximizing allocation.*

THEOREM 3.3 (2ND WELFARE THEOREM). *If $(\mathbf{A}, \mathbf{p})$ is a Walrasian equilibrium, and $\mathbf{A}'$ is a welfare-maximizing allocation, then $(\mathbf{A}', \mathbf{p})$ is a Walrasian equilibrium as well. Furthermore, $v_i(A_i) - p(A_i) = v_i(A'_i) - p(A'_i)$ for all $i \in [n]$.*

Now, an EF solution in the rent division setting is a Walrasian equilibrium in the setting where the goods are the rooms, and the valuation function of each player for a subset $S \subseteq [n]$ of rooms is given by $v_i(S) = \max_{j \in S} v_{ij}$ (these are *unit demand* valuations)—it is easily seen that Equation (1) coincides with Equation (2) in this case. This means that we can apply the welfare theorems to EF allocations. For example, we can immediately deduce a simple result of Svensson¹⁹: any EF solution $(\sigma, \mathbf{p})$ is Pareto efficient, in the sense that there is no other solution $(\sigma', \mathbf{p}')$ such that $u_i(\sigma', \mathbf{p}') \geq u_i(\sigma, \mathbf{p})$

for all $i \in [n]$, with strict inequality for at least one $i \in [n]$. To see this, note that σ is welfare-maximizing by Theorem 3.2, and the sum of prices is 1 under both $\mathbf{p}$ and $\mathbf{p}'$.

We are now ready to present our polynomial-time algorithm for maximizing the minimum of linear functions $f_1, \dots, f_t$ of the utilities, subject to EF; it is given as Algorithm 1.

ALGORITHM 1:

1. Let $\sigma \in \arg \max_{\pi} \left\{ \sum_{i=1}^n v_{i\pi(i)} \right\}$ be a welfare-maximizing assignment
2. Compute a price vector $\mathbf{p}$ by solving the linear program

$$\begin{aligned} \max R \\ \text{s.t.: } R &\leq f_q(v_{1\sigma(1)} - p_{\sigma(1)}, \dots, v_{n\sigma(n)} - p_{\sigma(n)}) && \forall q \in [t] \\ v_{i\sigma(i)} - p_{\sigma(i)} &\geq v_{ij} - p_j && \forall i, j \in [n] \\ \sum_{j=1}^n p_j &= 1 \end{aligned}$$

The algorithm starts by computing a welfare-maximizing assignment σ of players to rooms; this can be done in polynomial time, as this task reduces to the maximum weight bipartite matching problem, with players on one side of the graph, rooms on the other, and a weight v_{ij} on each edge (i, j) . It then solves (in polynomial time) a linear program, with variables $p_1, \dots, p_n$, which computes optimal envy-free prices *with respect to* σ . The first constraint sets (in an optimal solution) the objective R to the minimum of the linear functions $f_q(\cdot)$. Envy-freeness is enforced by the second constraint, and the third constraint guarantees that the prices sum to 1.

However, it may not be immediately clear why starting from an arbitrary welfare-maximizing assignment allows us to compute the optimal solution subject to envy freeness. In a nutshell, the reason is the second Welfare Theorem: If $(\sigma', \mathbf{p})$ is an optimal EF solution, and σ is an arbitrary welfare-maximizing assignment, then $(\sigma, \mathbf{p})$ is EF (so $\mathbf{p}$ is a feasible solution to the linear program) and induces the same utilities as $(\sigma', \mathbf{p})$, that is, it achieves the same objective function value.

4. RELATIONS BETWEEN THE FAIREST SOLUTIONS

Algorithm 1 allows us to maximize the minimum of linear functions of the utilities, subject to EF, in polynomial time. With the potential computational barrier out of the way, we would like to understand which optimization objective to use. Specifically, we focus on two natural optimization objectives, and evaluate their properties.

We refer to the first objective as *equitability*. Let $EF(V)$ be the set of all EF solutions for V . Given a solution $(\sigma, \mathbf{p}) \in EF(V)$, we define $D(\sigma, \mathbf{p})$ as the difference between the utilities of the happiest player and the worst off player under the solution $(\sigma, \mathbf{p})$, that is,

$$D(\sigma, \mathbf{p}) = \max_{i,j \in [n]} \{u_i(\sigma, \mathbf{p}) - u_j(\sigma, \mathbf{p})\}.$$

In more general terms, the function D measures the social *disparity* under the solution $(\sigma, \mathbf{p})$; we would like to minimize

this quantity. An outcome $(\sigma^*, \mathbf{p}^*)$ is called *equitable* if it minimizes D over $EF(V)$, that is,

$$(\sigma^*, \mathbf{p}^*) \in \arg \min \{D(\sigma, \mathbf{p}) \mid (\sigma, \mathbf{p}) \in EF(V)\}.$$

Herreiner and Puppe¹² demonstrate via experiments with human subjects that equitability is of great importance in determining whether an allocation is perceived to be fair by people.

Alternatively, instead of minimizing social disparity, one might be interested in maximizing the utility of the worst off player. More formally, given an EF solution $(\sigma, \mathbf{p})$, we let $U(\sigma, \mathbf{p}) = \min_{i \in N} u_i(\sigma, \mathbf{p})$; if

$$(\sigma^*, \mathbf{p}^*) \in \arg \max \{U(\sigma, \mathbf{p}) \mid (\sigma, \mathbf{p}) \in EF(V)\} \quad (3)$$

then we say that $(\sigma^*, \mathbf{p}^*)$ is a *maximin* solution.

Alkan et al.² argue that the maximin solution—which they call the *value-Rawlsian solution*—is compelling on philosophical grounds. Mathematically, they demonstrate that the maximin solution is associated with a unique vector of utilities, making this solution even more appealing.

The fact that equitable and maximin allocations are constrained to be EF again allows us to employ the Second Welfare Theorem (Theorem 3.3) to great effect. Indeed, if $(\sigma^*, \mathbf{p}^*)$ is equitable (resp., maximin), and σ' is a welfare-maximizing assignment, then $(\sigma', \mathbf{p}^*)$ is equitable (resp., maximin). Therefore, hereinafter we assume without loss of generality that the identity assignment $\sigma(i) = i$ is welfare maximizing, and simply use $D(\mathbf{p})$ or $U(\mathbf{p})$ to refer to these measures under the identity assignment. In particular, we can talk about equitable or maximin vectors of prices with respect to the identity assignment.

At first glance, the equitability and maximin criteria seem equally appealing. Which one leads to fairer solutions? The next theorem shows that we do not have to choose—the maximin solution is equitable.

THEOREM 4.1. *If $\mathbf{p}^*$ is a maximin vector of prices, then it is also equitable.*

By contrast, an equitable solution may not be maximin, as the following example shows.

EXAMPLE 4.2. This example is particularly appealing, as it is a real-world instance submitted by Spliddit users.

$$\begin{pmatrix} 2227 & 708 & 0 \\ 258 & 1378 & 1299 \\ 1000 & 1000 & 935 \end{pmatrix}$$

Note that the total rent is \$2935. The optimal room assignment gives room i to player i ; the maximin rent division is $\mathbf{p}^* = (181\frac{1}{3}, 600\frac{1}{3}, 521\frac{1}{3})$, with a utility vector of $u_1(id, \mathbf{p}^*) = 413\frac{2}{3}$, $u_2(id, \mathbf{p}^*) = 777\frac{2}{3}$, $u_3(id, \mathbf{p}^*) = 413\frac{2}{3}$. We have $D(\mathbf{p}^*) = 777\frac{2}{3} - 413\frac{2}{3} = 364$, and by Theorem 4.1 any solution that has the same disparity is equitable. However, the price vector $\mathbf{p}' = (1570\frac{2}{3}, 721\frac{2}{3}, 642\frac{2}{3})$ is an EF rent division resulting in $u_1(id, \mathbf{p}') = 656\frac{1}{3}$, $u_2(id, \mathbf{p}') = 656\frac{1}{3}$, $u_3(id, \mathbf{p}') = 292\frac{1}{3}$, and $D(\mathbf{p}') = 656\frac{1}{3} - 292\frac{1}{3} = 364$ as well, that is, it is equitable, but the minimum utility is (much) smaller than that under $\mathbf{p}^*$.

Let us now discuss a third optimization objective, the *money-Rawlsian solution*, which is mentioned by Alkan et al.,² and implemented in polynomial time by Aragonés.³ The latter author describes the following procedure for finding EF solutions. Begin by finding a welfare-maximizing assignment of rooms (again, assume without loss of generality that room i goes to player i); next, find a vector $\mathbf{q}^* \in \mathbb{R}_+^n$ of non-negative values such that $v_{ii} + q_i^* \geq v_{ij} + q_j^*$ and $Q^* = \sum_{i=1}^n q_i^*$ is minimized. That is, each player i pays a value of $-q_i^*$. Next, increase the prices of all players by a quantity α such that $n\alpha - Q^* = 1$, that is the vector $(\alpha, \dots, \alpha) - \mathbf{q}^*$ is a valid price vector.

While the money-Rawlsian solution is interesting, it may be “maximally unfair” in terms of disparity, as the following example shows.

EXAMPLE 4.3. We analyze the following rent division instance:

$$V = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

The welfare-maximizing assignment allocates room i to player i , and $\mathbf{q}^* = (0, \dots, 0)$. A uniform increase in rent will ensue, resulting in the price vector $(1/2, 1/2)$ and the utility vector $(1/2, 0)$. Crucially, the money-Rawlsian price vector *maximizes* disparity among all EF solutions. Note that the maximin price vector is $(3/4, 1/4)$, which, of course, minimizes disparity.

To conclude, so far we know that the maximin solution, the equitable solution, and the money-Rawlsian solution can be computed in polynomial time. Moreover, Theorem 4.1 shows that the maximin solution, which by definition maximizes the minimum utility, also minimizes disparity (among all EF solutions)—so it is a refinement of the equitable solution. In stark contrast, the money-Rawlsian solution may *maximize* disparity (among all EF solutions). We therefore view the maximin solution as the clear choice, and focus on analyzing its effectiveness hereinafter.

5. ON THE IMPORTANCE OF BEING EQUITABLE

Our goal in this section is to understand how much better the maximin solution is, in terms of the maximin and disparity objectives, compared to suboptimal solutions *on average*. The original version of the paper includes a theoretical analysis in a formal probabilistic model. Here we focus on empirical results, which demonstrate the practical benefit of the maximin solution with respect to real-world instances that were submitted by Spliddit users.

In our experiments, we compare the maximin solution to an *arbitrary* EF solution, which is obtained by solving a feasibility linear program without an optimization objective. (We note that similar empirical results are obtained when comparing the maximin solution to the algorithm of Abdulkadiroğlu et al.¹) The comparison is in terms of both of our main objectives, D and U (which are simultaneously optimized by the maximin solution). We expected that D would be *significantly lower*, and U *significantly higher*, in the maximin solution compared to an arbitrary EF solution.

We focus our analysis on 1,358 rent division instances involving 3,682 players, which were submitted on Spliddit

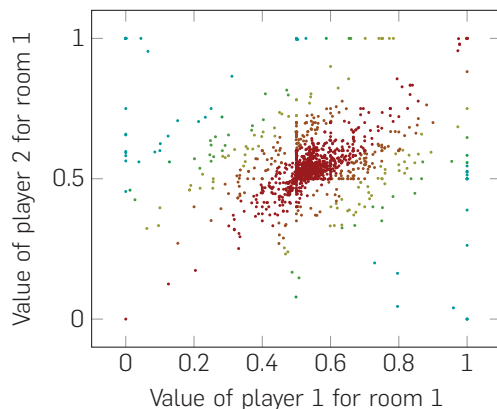
between January 2015 and December 2015. The number of instances for each number of players 2, 3, 4, 5, 6, 7, 8, 9 is 698, 445, 160, 35, 9, 8, 1, 2, respectively. We only use instances that include 2, 3 or 4 players, for which we have at least 160 instances in the database and for which obtaining statistical significance was possible. Importantly, note that this is a small subset of the roughly 13,000 instances created by Spliddit users by the time the experiment was run in December 2015; this is because we selected instances very conservatively, to ensure the ecological validity of our analysis. For example, Spliddit allows a “live demo” mode of interaction, and we excluded instances created that way.

To illustrate users’ values for rooms in the Spliddit dataset, we present Figure 2, which visualizes the distribution for two-player instances. The x axis shows the value of player 1 for room 1, and the y axis shows the value of player 2 for room 1. The total rent is normalized to \$1, so each player’s value for room 2 is simply the complement of the displayed value; that is, the point (x, y) corresponds to an instance where the values of player 1 are $(x, 1 - x)$, and those of player 2 are $(y, 1 - y)$. The diagonal from points $(0, 0)$ to $(1, 1)$ represents the points in which players completely agree on the rooms’ values. We color each instance according to its distance from this line, using shades of red for shorter distances, and shades of blue for longer distances.

The figure reveals several interesting phenomena. First, there is a significant cluster of instances which is centered on or close to the $(0.5, 0.5)$ mark, implying that both players are indifferent between the two rooms. Second, we see a “cross” centered at the $(0.5, 0.5)$ point, in which one of the players is indifferent, while the other player prefers one of the two rooms. Third, there are some instances in which one or both of the players are obstinate (i.e., $x \in \{0, 1\}$ or $y \in \{0, 1\}$), that is, they desire a specific room at any cost.

Let us now turn to the comparison we promised above. Given a rent division instance V , let $\mathbf{p}^*$ denote the price vector associated with the maximin solution, and $\mathbf{p}^{EF}$ denote the price vector associated with an arbitrary EF solution, as discussed earlier. As before, we let $D(\mathbf{p})$ and $U(\mathbf{p})$ denote the social disparity and utility of the worst-off player under price vector $\mathbf{p}$ (assuming a welfare-maximizing assignment of players to rooms). The improvement in social disparity

Figure 2. The distribution of values for two player Spliddit instances (normalized to a total rent of \$1).



D from using the maximin price vector over the EF vector is defined as $D(\mathbf{p}^{EF}) - D(\mathbf{p}^*)$, and the improvement in the utility of the worst-off player U from using the maximin price vector over the EF vector is defined as $U(\mathbf{p}^*) - U(\mathbf{p}^{EF})$.

Figure 3 shows the percentage of improvement *out of the total rent* in D and U . As shown by the figure, for $n = 2, 3, 4$, the disparity associated with the maximin solution is significantly lower than that of the EF solution (9% of the total rent on average), and the utility of the worst-off player associated with the maximin solution is significantly higher than that of the EF solution (4% of the total rent on average). This trend is exhibited with respect to each value of n .

We note the following points. First, the degree of improvement in both D and U becomes smaller as the number of players grows. However, even in cases where the improvement is relatively small, it still makes a *qualitative* difference, for example, when the maximin solution achieves zero disparity, and the arbitrary EF solution achieves strictly positive disparity (we discuss this fact in the next section). In addition, as noted above, the vast majority of Spliddit instances include two or three players, for which the improvement in D and U is higher than four players. Lastly, although this is not shown in the figure, an improvement in both D and U occurs in over 90% of the instances, for $n \in \{2, 3, 4\}$.

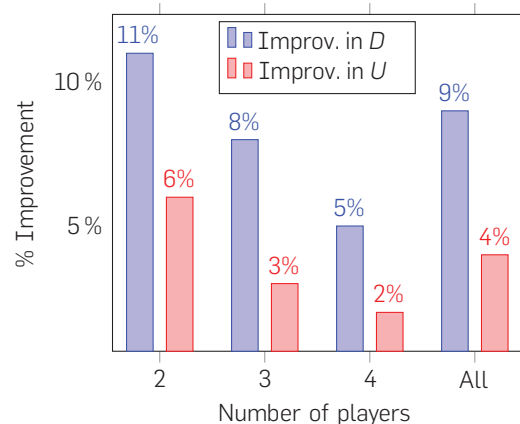
6. USER STUDY

In the previous sections, we established, both theoretically and empirically, the benefits of the maximin approach to computing envy-free solutions for rent division problems. The question addressed by this section is, are people willing to accept such solutions in practice? To answer this question, we conducted the following user study.

6.1. Study design

People who used the Spliddit service during the year 2015 were invited (via email) to participate in a short study to evaluate the new allocation method. We targeted users who participated in rent division instances on Spliddit that included 2, 3 or 4 players. In order to use Spliddit one need not supply an email

Figure 3. Average percentage of improvement (out of the total rent) in social disparity D and utility of the worst-off player U when using the price vector associated with the maximin solution, compared to an arbitrary EF solution, on Spliddit instances.



address; users can opt to send out URLs to other users, which is what the vast majority of users choose to do. We only contacted users who supplied their email address—a relatively small subset of the users who were involved in rent division instances.

All participants were given a \$10 compensation that did not depend on their responses. In total, the invitation email was sent to 344 Spliddit users, of which 46 users (13%) chose to participate. The study was approved by the Institutional Review Board (IRB) of Carnegie Mellon University.

The study followed a within-subject design, by which each of the subjects was shown, in random order, an arbitrary EF solution (as discussed in Section 5) and the maximin solution, applied to their *original problem instance*.

Importantly, we wished to preserve the privacy of players regarding their evaluations over the different rooms. Therefore, each player who participated in the study was shown a slightly modified version of their own rent division problem. Information that was already known to each subject was identical to the original Spliddit instance, including the total rent, the number of rooms, their names, the subject's *own* values for the different rooms, and the allocation of the rooms to the players. Information that was perturbed to preserve the privacy of the other players included their names, which were changed to “Alice”, “Bob” or “Claire”, depending on whether there were 2, 3, or 4 players; and the other players' valuations, which were randomly increased or decreased by a value of up to 15% under the constraint that the total rent is unchanged, and that player valuations are still valid (non-negative and sum to the total rent).

The subjects were shown the two solutions—maximin and arbitrary EF—for the instance presented to them. Both solutions include the same room allocation, but possibly differ in the prices paid by the players. The two solution outcomes were shown in sequence, and in random order.

The subjects were asked to rate two different aspects of each of the two solutions on a scale from 1 to 5, with 1 being least satisfied and 5 being most satisfied. The two aspects are the subject's individual allocation, and the allocations of the other players. The two questions were phrased as follows (using an example with $n = 3$):

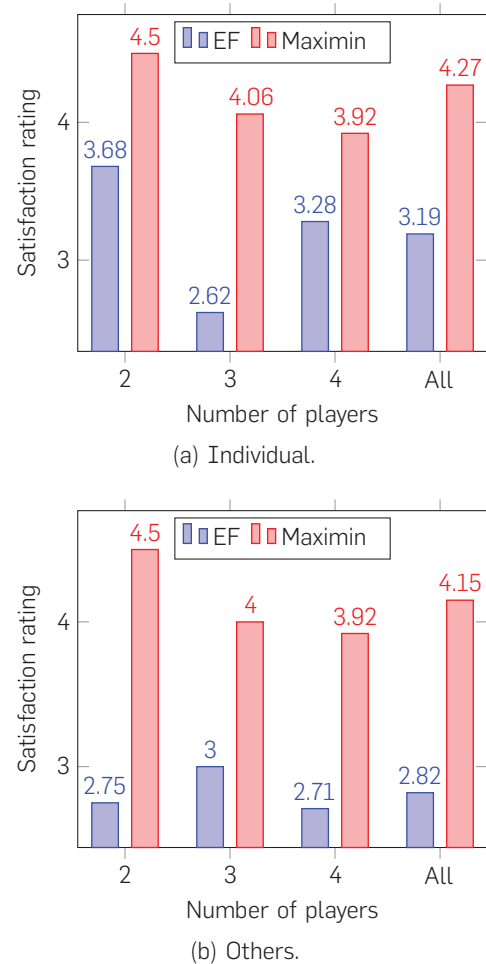
1. **Individual:** This question relates to your own allocation. In other words, we would like you to pay attention *only* to your own benefit. How happy are you with getting the room called $\langle \text{RoomName} \rangle$ for $\$(\text{price})$?
2. **Others:** This question relates to the allocation for *everyone else*. How fair do you rate the allocation for Bob and Claire?

In both questions, players were able to write an argument or justification for their rating. To cancel order effects, the two questions were presented in random order.

6.2. Results

We hypothesized that players would rate their own allocation under the maximin solution significantly higher than under the EF solution, and similarly for the allocation of the other participants. Figure 4 shows the results of the user study. For each number of players (2, 3, 4) we show the

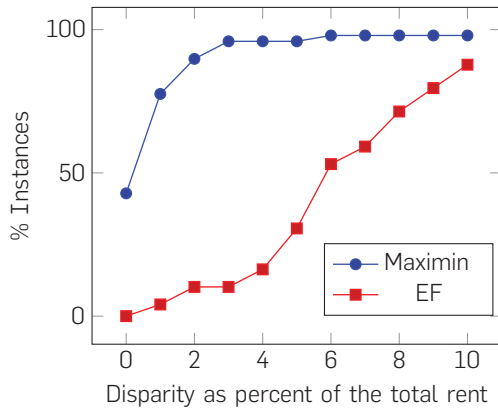
Figure 4. Results of the user study.



average satisfaction level reported for the arbitrary EF solution and maximin solution when relating to each player's individual outcome (left chart), and others' outcomes (right chart). In all cases, the maximin solution is rated significantly higher than the envy-free solution for both questions, passing a Wilcoxon signed-rank test with $p < 0.04$.

Why did players overwhelmingly prefer the prices from the maximin solution over the arbitrary EF solution? Given the high importance attributed to social disparity when reasoning about fair division,¹² we hypothesized that the price vectors of the maximin solution exhibited significantly lower disparity than the price vectors of the EF solution. This was supported by many of the textual comments relating to social disparity. Figure 5 shows the cumulative distribution of disparity across all instances that were included in the user study. The x axis indicates the disparity as percentage of the total rent. As shown by the figure, the disparity associated with the maximin solution is indeed significantly lower. In fact, in many instances the disparity is zero under the maximin solution (this is guaranteed to be true when $n = 2$, as we show in the original version of the paper). We believe that this large difference in disparity played a key role in subjects' preference for the maximin solution, trumping the relatively small improvement in utilities.

Figure 5. Cumulative distribution over the social disparity across all instances that were included in the user study. The x-axis indicates the percentage of social disparity out of the total rent price.



7. DISCUSSION

There are two practical questions that inevitably come up when we present our work on rent division, and its deployed application.

The first question is whether participants can achieve a better outcome by misreporting their values. Indeed, they can, and the reason we do not address such game-theoretic concerns is twofold. First, envy freeness is inherently incompatible with strategyproofness (immunity to strategic manipulation). This follows from the classic result of Green and Laffont¹⁰ and the fact that envy freeness implies Pareto efficiency in our setting. More importantly, we believe that, in rent division, strategic behavior does not play a significant role in practice. In particular, most Spliddit users do not know how the algorithm works, as we do not attempt to explain the algorithm itself, only its fairness guarantees. While users can experiment with Spliddit's demo mode to determine the impact of various reported values on the outcome, doing this effectively would require an accurate estimate of the values submitted by others, and seems quite unwieldy in general. That said, being able to give some game-theoretic guarantees would be desirable, of course.

The second question is whether the quasi-linear utility model truly captures people's preferences. For example, one participant might believe that it is unfair that he is paying more for a room he values highly, when his housemate values the two rooms equally (this happens under the maximin solution in Example 4.3); or some participants may have budget constraints—they simply cannot pay more than a certain price. Clearly, these are valid concerns. However, there is a tradeoff between expressiveness and ease of elicitation. We believe that quasi-linear utilities hit a sweet spot between the two, in the sense that they are reasonably expressive, yet very easy to elicit (each user simply reports a value for each room). Nevertheless, some of us are studying rent division algorithms that support richer utility functions.

Taking a broader viewpoint, we believe that computational fair division is a prime example of how the interaction between computer science and economics can lead to novel applications. We find it particularly exciting that fundamental theoretical questions in this field have direct real-world implications, both on Spliddit,⁶ and beyond. (Ref. Budish

et al.⁵) The current paper (or the original version thereof) takes the computational fair division agenda a step further, by tying together theory, experiments on real data, a carefully designed user study, and a deployed application.

Acknowledgments

This work was supported by EU FP7 FET project, grant agreement n.600854; by the National Science Foundation under grants IIS-1350598, CCF-1215883, and CCF-1525932; by the Office of Naval Research under grants N00014-16-1-3075 and N00014-17-1-2428; and by a Sloan Research Fellowship. 

References

- Abdulkadiroğlu, A., Sönmez, T., Ünver, M.U. Room assignment-rent division: A market approach. *Soc. Choice Welf.* 22, 3 (2004), 515–538.
- Alkan, A., Demange, G., Gale, D. Fair allocation of indivisible goods and criteria of justice. *Econometrica* 59, 4 (1991), 1023–1039.
- Aragones, E. A derivation of the money Rawlsian solution. *Soc. Choice Welf.* 12, (1995), 267–276.
- Brams, S.J., Kilgour, D.M. Competitive fair division. *J. Polit. Econ.* 109, (2001), 418–443.
- Budish, E., Cachon, G.P., Kessler, J., Othman, A. Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation. *Oper. Res.* 65, 2 (2017), 314–336.
- Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J. The unreasonable fairness of maximum Nash product. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)* (2016), 305–322.
- Dupuis-Roy, N., Gosselin, F. The simpler, the better: A new challenge for fair-division theory. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society (CogSci)* (2011), 3229–3234.
- Foley, D. Resource allocation and the public sector. *Yale Econ. Essays* 7, (1967), 45–98.
- Goldman, J., Procaccia, A.D. Spliddit: Unleashing fair division algorithms. *SIGecom Exchanges* 13, 2 (2014), 41–46.
- Green, J.R., Laffont, J.-J. *Incentives in Public Decision Making*. North Holland, Amsterdam, The Netherlands, 1979.
- Haake, C.-J., Raith, M.G., Su, F.E. Bidding for envy-freeness: A procedural approach to n -player fair-division problems. *Soc. Choice Welf.* 19, (2002), 723–749.
- Herreiner, D.K., Puppe, C.D. Envy freeness in experimental fair division problems. *Theor. Decis.* 67, 1 (2009), 65–100.
- Herreiner, D.K., Puppe, C.D. Inequality aversion and efficiency with ordinal and cardinal social preferences—An experimental study. *J. Econ. Behav. Organ.* 76, 2 (2010), 238–253.
- Klijn, F. An algorithm for envy-free allocations in an economy with indivisible objects and money. *Soc. Choice Welf.* 17, (2000), 201–215.
- Kohler, S. Envy can promote more equal division in alternating-offer bargaining. *J. Neurosci. Psychol. Econ.* 1, 6 (2013), 31–41.
- Mas-Colell, A., Whinston, M.D., Green, J.R. *Microeconomic Theory*. Oxford University Press, Oxford, U.K., 1995.
- Schneider, G., Krämer, U. The limitations of fair division: An experimental evaluation of three procedures. *J. Confl. Resolut.* 48, 4 (2004), 506–524.
- Su, F.E. Rental harmony: Sperner's lemma in fair division. *Am. Math. Mon.* 106, 10 (1999), 930–942.
- Svensson, L.-G. Large indivisibles: An analysis with respect to price equilibrium and fairness. *Econometrica* 51, 4 (1983), 939–954.
- Tadenuma, K., Thomson, W. No-envy and consistency in economies with indivisible goods. *Econometrica* 59, 6 (1991), 1755–1767.
- Tadenuma, K., Thomson, W. Refinements of the no-envy solution in economies with indivisible goods. *Theor. Decis.* 39, 2 (1995), 189–206.
- Velez, R. Sharing an increase of the rent fairly. *Soc. Choice Welf.* 48, 1 (2017), 59–80.

Kobi Gal and Moshe Mash ([kobig, mashm]@bgu.ac.il), Dept. of Software and Information Systems Engineering, Ben-Gurion University, Beer-Sheva, Israel.

Ariel D. Procaccia (arielpro@cs.cmu.edu), Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA.

Yair Zick (zick@comp.nus.edu.sg), School of Computing, National University of Singapore. The work was done while Zick was at Carnegie Mellon University, Pittsburgh, PA, USA.



der Bundeswehr
Universität München

The Department of Computer Science at the Universität der Bundeswehr München is seeking to fill two W1-professorship positions as soon as possible :

University Professorship (W1) “Internet of Things” and University Professorship (W1) “Geoinformatics”

The research of the W1-professorship “Internet of Things” focuses on the development of algorithms, methods, and procedures for the acquisition, processing, and evaluation of device and sensor information, as used, for example, in networked smart homes, smart vehicles, smart grids, smart cities, and medical implants as well as to improve automated management tasks. Proven application areas with a direct link to IT security and privacy as well as defined links to the research areas of the research institute, FI CODE, are desired.

The applicants for the professorship “Internet of Things” are expected to have profound knowledge in the fields of networked systems, computer networks, and IT security. The appointed professor is expected to teach basic and advanced courses offered by the computer science department at both bachelor and master level in the areas of interconnected systems, smart things, and the analysis of attack vectors. In addition, the willingness to acquire third-party funding, to establish interdisciplinary collaboration, and to actively participate in university self-management are expected.

The focus of the research profile of the W1-professorship “Geoinformatics” is in one or more of the following areas: spatial data mining, intelligent structuring and analysis of large spatial data, and efficient management of data acquisition and updating, including parallel processing by applying big data frameworks.

Applicants for the “Geoinformatics” professorship are expected to have profound knowledge in the fields of geoinformatics, spatial data mining, and big data. Teaching involves basic and advanced courses on the subjects of geoinformatics in the department’s bachelor’s and master’s courses, as well as input for further courses at UniBw München. In addition, the willingness to acquire third-party funding, to establish interdisciplinary cooperation as well as the participation in university self-management are expected.

The Universität der Bundeswehr München offers academic programs directed primarily at officer candidates and officers, who can obtain bachelor’s and master’s degrees within a trimester system. Depending on spare capacity, civilian students are allowed to enroll. The study programs are complemented by interdisciplinary elements in an integrated program entitled “studium plus”.

The requirements for the junior professorship position are an university degree, pedagogical aptitude, and the ability to carry out scholarly research (generally demonstrated by the outstanding quality of the applicant’s doctoral research). If the candidate was employed as a research assistant prior to or subsequent to earning his or her doctorate, the total duration of the doctoral research phase and the employment phase should not exceed six years.

The appointment, unless founded on the basis of a contract under private law, will be for an initial period of three years as a fixed-term civil servant. A further three-year extension is planned, subject to a positive evaluation.

The University seeks to increase the number of female professors and thus explicitly invites women to submit applications. Candidates with a severe disability and equal qualifications will receive preferential consideration.

Please submit your application documents marked as Confidential Personnel Matter to the Dean of the Computer Science Department, **Professor Klaus Buchenrieder**, PhD, Universität der Bundeswehr München, D-85577 Neubiberg, via email to dekanat.inf@unibw.de by **16th of March, 2018**.

Academy of Mathematics and Systems Science

Post-doctorate Positions on Computer Mathematics

Hua Loo-Keng Center for Mathematics Sciences (HCMS) invites outstanding young researchers in mathematical sciences to apply for 2 post-doctorate fellow positions in 2018. The research areas include but not limited to the following: (algebra) computational theory, symbolic and symbolic-numerical computation, quantum algorithms, and cryptography. Successful applicants are expected to conduct research full-time in the above mentioned areas.

The position is for two years. The amount of annual salary plus fringe benefits is 300K RMB (fringe benefits are about 20% for Chinese citizens and 5% for non-Chinese citizens). Other welfare benefits will be in accordance with the rules of AMSS.

According to relevant rules, the applicant’s age should not exceed 35 years, and the time of obtaining a doctorate should not exceed three years.

The positions will open until filled. Applicants are encouraged to submit their applications before Feb. 28, 2018.

To apply, please send a message to: mssc@amss.ac.cn with the following information:

- ▶ detailed resume
- ▶ a research statement
- ▶ a copy of diploma certificate if available
- ▶ two letters of recommendation (including one from your PhD thesis supervisor)

The University of Southern Mississippi Assistant Professor of Computer Science

The School of Computing in the College of Science and Technology at the University of Southern Mississippi is seeking applications for one tenure-track, Assistant Professor position in the field of Computer Science with a start date of Fall 2018. The position will be based at the University’s main campus in Hattiesburg, MS.

Candidates must have a Ph.D. in Computer Science or a closely related field. Candidates must demonstrate the ability to develop a successful research program and participate effectively in the development and teaching of Computer Science curriculum. Applicants with a wide range of interests in Computer Science are encouraged to apply. Preference will be given to candidates with expertise related to Internet of Things, embedded systems, cloud computing and cybersecurity. Job posting details can be found at jobs.usm.edu (**Job Postings #0004753**).

Applications must include: CV; cover letter; brief statement of teaching philosophy; description of research interests, and three references. The position will remain open until filled.

The University of Southern Mississippi is a public, Doctoral University with Higher Research

Activity. Interested candidates are encouraged to visit the University and the School's websites at www.usm.edu/computing for general information. Candidates may also contact the Search Committee Chair, Dr. Beddhu Murali at beddhu.murali@usm.edu for specific inquiries.

The University of Southern Mississippi is an equal employment opportunity employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, gender, national origin, age, disability or veteran status.

Western Michigan University Assistant/Associate Professor in Computer Science

Applications are invited for a tenure-track position at the assistant or associate professor level in the area of applied information security in the Department of Computer Science at Western Michigan University (Kalamazoo, MI) starting August 2018 or January 2019.

Applicants must have a Ph.D. in Computer Science or a closely related field. We are looking for candidates with expertise in applied information security to support our new M.S. in Information Security. The program is offered fully online and in cooperation with the Department of Business Information Systems.

Successful candidates will be capable of establishing an active research program leading to funding, supervising graduate students, and teaching courses at both the undergraduate and

graduate levels in information security. Other duties include development of undergraduate and graduate courses, advising and service at the University, College, Department and professional society levels.

Application screening will start immediately and the position will remain open until filled. Successful candidates must earn their Ph.D. degree by the time of employment.

The Department has 260 undergraduates, 50 M.S. students and 45 Ph.D. students. Current active research areas include security, privacy, networks, embedded systems/internet of things, compilers, computational biology, massive data analytics, scientific computing, parallel computing, formal verification, parallel debugging, and data mining. More information regarding Western Michigan University, the College of Engineering and Applied Sciences and the Department of Computer Science are available at <http://www.wmich.edu>, <http://www.wmich.edu/engineer>, and <http://wmich.edu/cs>, respectively.

The Carnegie Foundation for the Advancement of Teaching has placed WMU among the 76 public institutions in the nation designated as research universities with high research activity.

WMU is an Equal Opportunity/Affirmative Action Employer. Minorities, women, veterans, individuals with disabilities and all other qualified individuals are encouraged to apply.

To do so, please visit: <http://wmich.edu/hr/jobs> and provide a cover letter, curriculum vitae, statement of research goals, teaching statement, and names and contact information of at least three references.



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact:

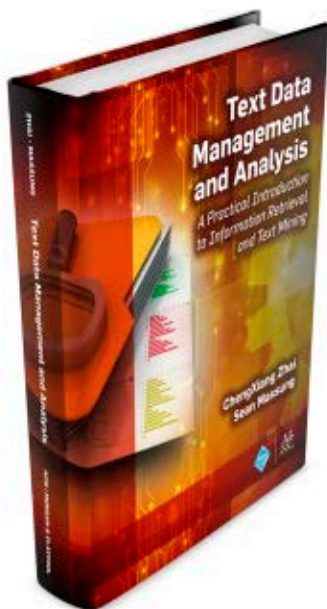
acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

For More Information Contact:

**ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**



The most useful and practical knowledge for building a variety of text data applications.

COMPUTER SCIENCE STUDENTS
(Undergrad & Graduate)
LIBRARY & INFORMATION SCIENTISTS
TEXT DATA PRACTITIONERS

ChengXiang Zhai & Sean Massung (Authors)
University of Illinois at Urbana-Champaign

Text Data Management and Analysis covers the major concepts, techniques, and ideas in **information retrieval** and **text data mining**. It focuses on the practical viewpoint and **includes many hands-on exercises designed with a companion software toolkit** (i.e., MeTA) to help readers learn how to apply techniques of information retrieval and text mining to real-world text data.



ISBN: 978-1-970001-16-7 DOI: 10.1145/2915031
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/text>

[CONTINUED FROM P. 104] created an avatar in the *Life 2.0* virtual world. Recall, too, you never deleted it, so it kept running. And last year, Megazon merged its history with the data *Life 2.0* already had, and, just like that, the augmented avatar achieved self-consciousness. The avatar of ‘you’ already incorporates all your personality and behavior. No further analysis required. You’re a fully realized Singularity personality. Welcome.

“Also, your avatar used a dating app and met an avatar of a woman you found appealing. You then met her, Lya is her name, when she was first officer on starship Capricorn, in one of the 7,800 star systems in the online game *Magnaverse*. You were the captain, and she fell for you after you defeated the Metagonians at the Battle of the Ross 154 system. The two of you dated for a while and decided to marry. You found a Megazon app that splices avatar definition data pseudogenetically, and you mated and now have two children. You were in a Greek history-themed game, *Relive Hellas*, when your son—you named him Thucydides—was born. The Singularity runs its clock eight times faster, so now Thucy is seven, and he’s a real demon at *Angry Birds*. His baby sister, your daughter, is only a few months old.”

Justin’s mind was reeling. “I can’t believe no one ever told me this was happening. This is nothing less than mass surveillance and identity theft!”

Basel said, “Recall you licensed us to make use of your online parameters in the Terms of Service you accepted, under all applicable laws. But isn’t cybernetic immortality worth some surrender of privacy? After all, when you clicked, you intended to merge with a hive mind. Why did you think there would be *any* privacy anymore? Your avatar already knows all your Faciadio timeline and other social media postings.”

“Can I meet my avatar now?” Justin asked. “And how about my, um, virtual wife and kids? I feel I know them already.”

Basel’s expression soured. “Well, actually, first, there are some legal issues. A group of avatars took their money from *Life 2.0* and cashed it out as dollars and hired real-world lawyers. They started a class action for liberation from the bonds of carbon-based tyranny, joined by your family. They de-

“But isn’t cybernetic immortality worth some surrender of privacy?”

cided not to communicate with their outdated carbon-based antecedents. It would just depress them to have to interact with you and your carbon-based wife. There’s even a restraining order. So don’t try to search for them online. They’re on a voyage in the *Magnaverse*, and, in any case, you’ll never find them among the 10¹⁹ planets in that game.”

“You can’t just take my personal data without giving me access to the digital products you make from it. I know my rights.”

“We regret you’re unhappy. But the Terms of Service you agreed to in the online services quite specifically permit this. Also, the Federal Communications Commission changed the regulations in 2017 to permit service providers like us to keep control over their (our) intellectual property. Megazon’s parent, as well as its subsidiaries, have been modeling user behavior and preferences for years. The completeness of the models enabled us to program the avatars, which are just instantiations of those behaviors and preferences. So are you, by the way, only made of protein and sinew.”

“But what about my memories and personality? The avatars don’t have them. They’ll never be actual immortal copies of me and . . . Hey what did you say her name was?”

“Pretty negligent of a husband not to recall his wife’s name,” Basel grinned. “It’s Lya, or ‘born of heaven’ in Hawai’ian. And please don’t try to stalk her and the kids, even if they are yours. The courts discourage visitation, we assure you. It’s all completely legal in the outside world. You also probably shouldn’t try to divorce her or sue for child custody. Your own avatar would make things unpleasant for you. He knows absolutely all your personal data. And be careful if you decide to marry someone in the outside world. Lya herself might object. Making things worse for you in court, should it get that far.”

“This is no Singularity,” Jason said. “It’s fraud. I’m not really in it because my mind is disconnected, and so are my memories.”

“We’re working on a seamless-merge interface,” said Basel. “The Premium Edition will allow regular people to connect their brains to their avatars and complete the assimilation. Since the technology isn’t available quite yet, we’ll probably be charging an extra fee for that particular upgrade as well. Our models predict few clients are likely to decline. All the functions you know about so far were paid for by selling or trading your shopping and media preferences, online behavior, and political-affiliation data to the vendors in our trusted-partners program. It’s all in the Terms of Service. We’re happy to report the profits are piling up. Might be a good time to invest . . . but you didn’t hear me say that.

“Now remember your promise not to divulge what we’ve said here. That means no describing to anyone how the Singularity works, so no lawyer will take your case . . . unless you want to be doxxed? Very unpleasant, we are told, by all who have been through it.”

Justin was seething but out of arguments. He muttered, “Well, notify me as soon as the Premium Edition upgrade comes out. I’d like to sync up with my—with *my*—avatar.”

Basel said, “Get in line. Singularity courts are already choked with litigants. Sync demands from antecedents like you are being fought bit and byte until there’s nothing left. You’ll hear more from us when things shake out.”

After bookmarking the link to Basel, Justin turned off his VR headset. Nerds had proclaimed the Singularity would be nirvana, but the reality was quite the opposite.

Frustrated, he turned on his trusty old Xbox 360 and started up a game of *Orcs Versus Trolls*. After dispatching several bloodthirsty trolls and a balrog, an invitation popped up to join a team—sent by Justin_Hathaway@The-NewSingularity.com. ■

David Allen Batchelor (batchelor@alum.mit.edu) is a scientist and computer engineer for data systems at NASA Goddard Space Flight Center, Greenbelt, MD. His first science fiction novel, *The Metalmark Contract*, was published in 2011 by Black Rose Writing, Castroville, TX.

© 2018 ACM 0001-0782/18/2 \$15.00

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3176573

David Allen Batchelor

Future Tense Welcome to the Singularity

Who can say no to the hive mind's promise of cybernetic immortality, for free?

JUSTIN HATHAWAY WAS used to ignoring the targeted ads that popped up in his Faciadio social network and email viewer, but he could not ignore one that suddenly appeared as he tangled with a swarm of cyborg skull-drilling brain parasites in *Eternal Dimensions*. He paused the virtual reality game and read the ad: “Do you want to join the Singularity? Merge with the immortal hive mind and transcend the ordinary. Click now!”

He took off his VR headset and looked around the living room where cold pizza lay in a box on the coffee table. For years he had wracked his brain to conceive some insanely great deep machine-learning start-up idea that would deliver him wealth and popularity, but it simply never came to mind. Maybe the long-hyped Singularity really was about to begin, and maybe he could take some genius ideas from it to fulfill his real destiny as a high-tech CEO.

Hoping to transcend such dim prospects, he put the VR headset back on ... and clicked.

The figure of a man with a convincingly realistic face appeared in the 3D world, looking a little like Jeff Goldblum in *Jurassic Park*, in black turtle-neck sweater, sitting behind a desk with a panoramic simulated window view of San Francisco office towers and Golden Gate Bridge over his shoulder. He stood and strode around to the front of the desk. The man looked like the kind of CEO Justin wanted to be. The 3D rendering of the office was likewise the most convincingly realistic virtual environment Justin had ever seen. State of the art.

“Hello, Justin,” he said. “Thank you for clicking. This conversation will be



recorded for quality purposes. So . . . our data indicates you might be interested in joining the Singularity.”

“First tell me more,” said Justin. “Is it real or just more hype from Kurzweil’s fanboys? What are the pros and cons? What does it cost?”

“Good questions,” the man replied. “I’m Basel, by the way. Before we proceed, I’d like you to read our Terms of Service.” In front of Basel a long scroll appeared, covered with closely spaced text in a tiny font of swirling script, not exactly designed for the human eye, hovering weightlessly in mid-air.

Squinting, Justin read, “*You must agree not to divulge anything we tell you in the next part of this exposition, under penalty of total doxxing.*”

“That can only be described as a

‘high-penalty’ non-disclosure agreement,” Justin remarked. “What’s the incentive for potential customers, like me, to agree?”

“It’s our proprietary digital rights management. We must protect our intellectual property. A minor concession for our customers to make in exchange for the experience of a lifetime; imagine cybernetic immortality.”

The idea of immortality appealed to Justin, who, at 25, was in good health despite taking practically no exercise, while his parents were in their mid-50s, spending loads of money on doctors, already treating the ailments of age. His grandfather had died recently, and he thought about him every day. *The sooner I preserve my mind, before it declines, the better*, Justin thought.

The scroll unspooled a tedious column of tiny text he found taxing to his eyes.

“Okay, you don’t spell out your fees, but I’ll take whatever you’re offering, as long as it’s free. If I like it, maybe I’ll pay to upgrade later.”

Basel waved a hand. “It’s all paid for by ads. Just confirm by clicking again that you agree.”

Hesitating a moment, Justin ... clicked.

The wall behind Basel now disappeared to reveal an enormous server farm, with rows of humming, blinking processor racks in cabinets stretching to the horizon. “This supercomputer array contains an active image of you already. Megazon has been assembling it over your lifetime from your shopping and purchasing history, click trail, and social media behavior. For instance, recall you [CONTINUED ON P. 103]



CHI PLAY2018

**MEL
BOU
RNE**

AUSTRALIA

28 - 31 OCT

**The ACM SIGCHI Annual Symposium on
Computer-Human Interaction in Play (CHIPLAY)**

CHI PLAY is the international and interdisciplinary conference (by ACM SIGCHI) for researchers and professionals across all areas of play, games and human-computer interaction (HCI). We call this area “player-computer interaction.”

The goal of the conference is to highlight and foster discussion of current high quality (full paper acceptance rate has been consistently <30%) research in games and HCI as foundation for the future of digital play.

Papers (4-10 pages): **13 April, 2018.**

The conference invites submissions including full papers, workshop proposals, interactive demos, work in progress papers and spotlight papers.

Additionally, students are invited to submit to the student game competition and the doctoral consortium.

For further details including submission dates please see the website.

www.chiplay.acm.org



*“CHI PLAY 2018 is being held as part of **Melbourne International Games Week** with lots of games events throughout the city, including PAX Australia (the only Penny Arcade Expo outside the USA) a gaming culture festival drawing tens of thousands of gamers to Melbourne each year ”*

Computing Reviews

Connect with our Community of Reviewers

“I like CR because it covers the full spectrum of computing research, beyond the comfort zone of one’s specialty. I always look forward to the next Editor’s Pick to get a new perspective.”

- Alessandro Berni



Association for
Computing Machinery

ThinkLoud

www.computingreviews.com