READ YOUR FAVORITE PUBLICATIONS YOUR WAY

Now, your IEEE Computer Society technical publications aren’t just the most informative and state-of-the-art magazines and journals in the field — they’re also the most exciting, interactive, and customizable to your reading preferences.

The new myCS format for all IEEE Computer Society digital publications is:

- **Mobile friendly.** Looks great on any device — mobile, tablet, laptop, or desktop.
- **Customizable.** Whatever your e-reader lets you do, you can do on myCS. Change the page color, text size, or layout; even use annotations or an integrated dictionary — it’s up to you!
- **Adaptive.** Designed specifically for digital delivery and readability.
- **Personal.** Save all your issues and search or retrieve them quickly on your personal myCS site.

You’ve Got to See It!

To really appreciate the vast difference in reading enjoyment that myCS represents, you need to see a video demonstration and then try out the interactivity for yourself. Just go to [www.computer.org/mycs-info](http://www.computer.org/mycs-info).
MEASURING THE INTERNET
2016–2017 Editorial Calendar

Network Function Virtualization (Nov/Dec 2016)
Network function virtualization (NFV) — the practice of decoupling network hardware and software to allow network services to run on commodity cloud computing-style platforms — is a transformational vision that has taken the telecommunications industry by storm. Much in the same way it did for traditional IT, the hope is that NFV will foster innovation in the telecommunication industry by enabling faster deployment of new services with less risk.

ICT for Smart Industries (Jan/Feb 2017)
Developments in information and communication technology (ICT) for smart industries lead to a multitude of Internet-related research questions. Questions range from the design and analysis of sensor nodes and networks to data acquisition and machine-learning algorithms, including feedback control and optimization and cloud-based services. Orthogonal to these stand questions related to overall scalability, dependable security, data integrity, and privacy — as well as questions about sustainability.

Fog Computing (March/April 2017)
The Internet has witnessed two radical changes in the past decade: rapidly growing cloud computing and pervasive mobile devices. Despite many unresolved issues, cloud computing has quickly become essential to both enterprises and personal end users. Meanwhile, mobile devices (such as sensors, smartphones, and tablets) have become pervasive and are driving the development of many new applications across diverse domains — from transportation to healthcare to manufacturing to smart cities to smart grids — powered by ever-improving wireless networking and mobility support. Enabling this future Internet of Things imposes unique challenges. For example, many devices will have limited battery power and processing capabilities, and hence can’t support computational-intensive tasks. To this end, a new computing paradigm, fog computing, has emerged to distribute advanced computing, storage, networking, and management services to the edge of the network, close to the end users, thus forming a distributed and virtualized platform.

Usable Security (May/June 2017)
People are a vital part of any computing system, but they also frequently create security vulnerabilities and challenges for technology designers. This special issue of *IEEE Internet Computing* focuses on the design and understanding of security and privacy technologies (and the people who use them) by including articles based on work presented at the Symposium on Usable Privacy and Security. These articles will highlight the top results from the last two years, updated for the *IEEE Internet Computing* audience.

Energy-Efficient Data Centers (July/Aug 2017)
The advent of mega-scale Internet services and public cloud offerings led to a redesign of data center architectures, which addressed key inefficiencies, particularly in electrical and mechanical infrastructure. At the same time, the accelerated need for efficient servers spurred a generation of research on CPU, memory, network, and storage power-management techniques, which has led to a marked improvement in server efficiency and energy proportionality. However, it’s time for a second, holistic, clean-slate redesign of the data center, encompassing new server architectures, heterogeneous computing platforms, radical networking paradigms, new mechanical and electrical designs, intelligent cluster management, and radically rethinking software architectures while considering changing use patterns.
Energy-Efficient Data Centers
(July/August 2017)

Final submissions due: 28 October 2016

Please email the guest editors a brief description of the article you plan to submit by 28 September 2016.
Guest Editors: Weisong Shi and Thomas Wenisch (ic4-2017@computer.org)

In the last decade, data centers have become the core of modern business environments as computation has moved rapidly into the cloud. Data centers are among the fastest-growing users of electricity in the US, consuming an estimated 91 billion kilowatt-hours of electricity in 2013. They’re projected to increase to roughly 140 billion kilowatt-hours annually by 2020—the equivalent annual output of 50 power plants, costing American businesses $13 billion annually in electricity bills, and emitting nearly 100 million metric tons of carbon pollution per year. When operating a data center of hundreds of thousands of servers, it’s essential that they be operated effectively, to improve energy efficiency and environmental sustainability. With the aggressive adoption of cloud-based computing, the demands on data centers are growing exponentially, and both academia and industry will need to rethink how data centers are designed, built, and operated to be sustainable.

Despite a decade of research and industrial innovation, a recent Natural Resources Defense Council (NRDC) report indicates that typical small and midsize data centers hosting private clouds still have many wasteful practices. While best practices at mega-scale commercial cloud operators (such as Facebook, Microsoft, Google, and Amazon) have addressed the most egregious wastes (for example, inefficient cooling), we nevertheless must find ways to transfer these best practices across the data center landscape and address the remaining performance and efficiency challenges that afflict even the largest installations.

Around the mid-2000’s, the advent of mega-scale Internet services and public cloud offerings led to a redesign of data center architectures, which addressed key inefficiencies, particularly in electrical and mechanical infrastructure. At the same time, the accelerated need for efficient servers spurred a generation of research on CPU, memory, network, and storage power-management techniques, which has led to a marked improvement in server efficiency and energy proportionality. However, this first generation of improvement has plateaued; further opportunity in the large-scale mechanical infrastructure is limited, and no single server or network component stands out as the key source of inefficiency. Hence, it’s time for a second, holistic, clean-slate redesign of the data center, encompassing new server architectures, heterogeneous computing platforms, radical networking paradigms, new mechanical and electrical designs, intelligent cluster management, and radical rethinking of software architectures while considering changing use patterns (such as hybrid private/public clouds).

With this in mind, this special issue calls for research on various issues and solutions that can enable energy-efficient data centers. Topics of interest include (but aren’t limited to) the following:

- energy-efficient networks for data centers;
- energy-efficient virtualization techniques;
- instrumentation, measurement, and characterization studies;
- metrics, benchmarks, and interfaces;
- performance, energy, and other resource trade-offs, as well as energy complexity;
- energy-efficient software optimization and application design;
- system-level optimization and cross-layer coordination;
- scheduling, runtime adaptation, and feedback control;
- processor, memory, network, storage, hardware components, and architecture;
- reliability and power management;
- thermal management;
- green energy sources and their implications;
- technologies for and management of energy storage; and
- lifecycle analysis.

Submit a manuscript on ScholarOne at https://mc.manuscriptcentral.com:443/ic-cs
DEPARTMENTS

Standards
54 KaaS: A Standard Framework Proposal on Video Skimming
Lanshan Zhang, Linhui Sun, Wendong Wang, and Ye Tian

Internet Governance
60 What Is Algorithm Governance?
Danilo Doneda and Virgilio A.F. Almeida

Internet of Things, People, and Processes
64 From the Service-Oriented Architecture to the Web API Economy
Wei Tan, Yushun Fan, Ahmed Ghoneim, M. Anwar Hossain, and Schahram Dustdar

Natural Web Interfaces
69 Natural Interaction for Bot Detection
Robert St. Amant and David L. Roberts

Linked Data
74 Semantic Filtering for Social Data
Amit Sheth and Pavan Kapanipathi

COLUMNS

From the Editors
4 Customizing and Sizing the Internet for IoT Devices
M. Brian Blake

The Digital Citizen
79 From Zero to Hero: How Zero-Rating Became a Debate about Human Rights
Linnet Taylor

Peering
84 Emergent Collectives Redux: The Sharing Economy
Charles Petrie

Backspace
87 On Risk
Vinton G. Cerf

01 Call for Papers
25 IEEE Computer Society Info
59 Advertiser Index
As the Internet grows in size, complexity, and the role it plays in modern society, measuring the Internet is increasingly critical to guide its continued evolution. Yet the scale, diversity, opacity, and ethical implications of conducting Internet experiments make it difficult to obtain an accurate and representative understanding of the network’s behavior.
Customizing and Sizing the Internet for IoT Devices

M. Brian Blake • Drexel University

As a college student in 1991, the Internet was a different experience. At a summer fraternity conference in Washington, DC, I remember meeting a fraternity member from another chapter, and we decided to communicate by e-mail. At the time, I couldn’t remember my e-mail address, so I took his mailing address (because long-distance phone charges were much more expensive than a 12-cent stamp at the time) to send him a note with my e-mail address.

After sending the letter, I remember going to the computer lab several weeks later at my alma mater, Georgia Tech, to see if the message had arrived. When I finally remembered how to log in, I found that I had his message in addition to six other messages that came in over the past two months. All of those messages were sent directly to me. My email box was completely free of any unsolicited messages or spam. My fraternity brother’s first message was a simple one-liner test message, so I replied with a slightly longer message, to test my ability to send a reply message. Three days later, I returned to the lab, but with no response. When I returned two days later, he had sent a much longer message with details about their fraternity chapter and ways that we could collaborate.

Let’s compare that to now: I just checked my iPhone, which has been buzzing since I started writing this article, and I received 12 messages in the past 10 minutes. Usually, four of those messages are advertisements, six messages require my action from work, and two messages are from friends or family via social networking. Several messages have pictures. Other messages have links to Internet-based information and postings.

Clearly, over the past 25 years, traffic on the Internet has expanded in scope and in scale. What’s the future for the Internet with respect to usability? How will that affect size and scope? The Internet of Things (IoT) has been given a great deal of attention over the past five years. Research projects surrounding IoT tend to suggest that devices will interact over the Internet and co-exist with humans. In some way, this would require highly specialized devices to have the ability to customize diverse and open information into data nuggets that are useful for their operations. It occurs to me that we might need to develop a protocol underneath Web protocols that’s safe for device communication. However, we might be able to leverage contributions from other areas, including the following.

Normal forms for IoT. Normalization in a relational database naturally reduces redundancy, but allows for a structure that’s easier to maneuver. This challenge of disambiguating Web information for use in devices could be compared to the normal forms in relational databases. What if certain Web locations or specific communication protocols could be classified by a specific normalization level that relates to a specific type of device?

Engineering 4+1 views for IoT. Another method in software engineering also suggests the ability to create a view that specifically isolates a subset of information customized for a particular stakeholder. The idea of a 4+1 architectural view model, in software engineering, defines a system with multiple views (logical, development, process, and physical) where a fifth view or the +1 view connects all the others. Deriving a 4+1 view paradigm for the Internet might suggest multiple views associated with the varying dimensions of information available, but with development of a specific +1 view that directs usability for a specific class of IoT devices.
XSL transformations for IoT. Although a relatively dated technique, Extensible Stylesheet Language is a scripting language that allows an XML-based document to be translated into almost any other text-based format. There’s a parallel here, where Web-based information can be translated into a new language that’s specific to a class of devices. This transformation can remove redundant or unnecessary information.

It’s interesting to imagine how innovations in the fields of database management, software engineering, and Web applications might be redeployed to create a dimension within the Internet that’s a safe space for devices. As the scope of information expands on the Internet, it will be important to understand how humans, devices, or things might interact together efficiently and effectively.

This month’s special issue particularly addresses how we understand the Internet’s size and, in a sense, indirectly addresses the need to accommodate all the various stakeholders. I would like to thank the guest editors — Michael Rabinovich and Mark Allman — for the current special issue on “Measuring the Internet.” Moreover, I thank the authors for their articles, the reviewers for their service, and our readers for their interest. I hope that you enjoy the issue.

M. Brian Blake is the provost and executive vice president of academic affairs at Drexel University. As a professor of computer science and electrical engineering, his research interests are in service-oriented computing, adaptive distributed systems, and Web-based software engineering. Blake has a PhD in information and software engineering from George Mason University. Contact him at mbrian.blake@drexel.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

Customizing and Sizing the Internet for IoT Devices

Editor in Chief
M. Brian Blake • M.Brian.Blake@drexel.edu

Associate Editors in Chief
Barry Leiba • barryleiba@computer.org
Anirban Mahanti • anirban.mahanti@nicta.com.au
George Pallas • gpallas@cs.urey.ac.cy

Columnists
Backspace
Vinton G. Cerf • vger@cs.com
Digital Citizen
Kieron O’Hara • kmo@ecs.soton.ac.uk
From the Editors
M. Brian Blake • MBrian.Blake@drexel.edu
Peering
Charles J. Petrie • petrie@cs.drexel.edu
Practical Security
Hilary Orman • hilary@purplestreak.com

Department Editors
Beyond Wires
Yih-Farn Robin Chen • Chen@research.ait.com
Big Data Bits
Jimmy Lin • jimmylin@uwaterloo.ca
Developing World
Kamal Bhattacharya • kamal@k.com
Internet Governance
Vinglio Almeida • vinglio@dec.ufmg.br
Internet of Things, People, and Processes
Schahram Dustdar • dustdar@dsg.tuey.technion.ac.at
Linked Data
Carole Goble • cgp@cs.man.ac.uk
Natural Web Interfaces
Munindar P. Singh • singh@ncsu.edu
Spotlight
Gustavo Rossi • gustavo@ilis.info.unlp.edu.ar

Standards
Yong Cui • cuiyong@tsinghua.edu.cn
View from the Cloud
George Pallas • gpallas@cs.urey.ac.cy

Additional Editorial Board Members
Elena Bertino • elentina@ciena.purdue.edu
Fabian Bustamante • fabian@cs.northwestern.edu
Fred Douglass • fredouglass@computer.org
Stephen Farrell • stephen@caesar.freeservers.org
Elena Ferrari • elena.ferrari@unimi.it
Robert E. Filman • rfilman@computer.org
Michael N. Huels • huels@cs.ucsc.edu
Anu Huang • anu@cs.uu.chem
Anne-Marie Kermarrec • anne-marie.kermarrec@inria.fr
Peter Mika • pmika@yahoocom
Dejan Milojicic • dejan@hpl.hp.com
Michael Robinovich • michael.robinovich@case.edu
Amol Shelte • amol@cs.wright.edu
Weisong Shi • weisong@wayne.edu
Maarten van Steen • m.vansteen@utwente.nl
Craig W. Thompson • cwt@jhu.edu
Steve Vinosi • vinosi@iiee.org
* EIC emeritus

CS Magazine Operations Committee
Forest Shull (chair), M. Brian Blake, Maria Ebling, Lieven Eckhoff, Miguel Encarnacao, Nathan Ensmenger, Santi Helal, San Murugesan, Yong Rui, Ahmad-Reza Sadeghi, Dioniidis Spinellis, George K. Thiruvathukal, Mazin Yousef, Daniel Zeng

CS Publications Board
David S. Ebert (VP for Publications), Alfredo Benso, Irene Bojanova, Greg Byrd, Min Chen, Robert Dupuis, Niklas Ermoljev, Dovide Falesi, William Ribarsky, Forrest Shull, Melanie Tory

Staff
Editorial Management: Tammi Titworth, Manager, Editorial Services Content Development: Brian Brannon, bbbrannon@computer.org
Publications Coordinator: internet@computer.org

Director, Products & Services: Evan Butterfield
Senior Manager, Editorial Services: Robin Baldwin
Senior Business Development Manager: Sandy Brown
Advertising Coordinator: Deborah Sims, dsiims@computer.org

Technical cosponsor:

IEEE Internet Computing
IEEE Computer Society Publications Office
10862 Los Vaqueros Circle
Los Alamitos, CA 90720 USA

Editorial. Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author’s or firm’s opinion. Inclusion in IEEE Internet Computing does not necessarily constitute endorsement by IEEE or the IEEE Computer Society. All submissions are subject to editing for style, clarity, and length. Submissions. For detailed instructions, see the author guidelines (www.computer.org/internet/author.html) or log onto IEEE Internet Computing’s author center at ScholarOne (https://mc.manuscriptcentral.com/cs-ire). Articles are peer reviewed for technical merit. Letters to the Editors. Email lead editor Brian Brannon, bbbrannon@computer.org

Subscription Change of Address. Send requests to address.change@ieee.org.
Missing or Damaged Copies. Contact help@computer.org.
To Order Article Reprints. Email internet@computer.org or fax +1 714 821 4010.
At the heart of the Internet’s unquestionable success is simplicity and flexibility that not only facilitates easy communication, but also fosters innovative applications. However, as innovation drives the Internet into ever-deeper corners of our everyday lives, the technological ecosystem underlying our Internet use becomes increasingly complex. To continue the evolution of the Internet requires a sound and accurate understanding of how the network works, making the topic of this special issue — Internet measurement — a crucial component of advancing the state of networking.

Research and operational communities have made many advances in our understanding of the Internet and networking through a multitude of measurement efforts over the years. While advances will continue, we identify three key challenges that empiricalists are increasingly facing: scale, opacity, and ethical issues. These obstacles represent key areas where new methodologies and approaches are crucially needed.

Scale
The Internet is rapidly expanding along many axes, including users, businesses, devices, houses, criminals, applications, connection technologies, protocols, and threats. The immense scale means that the system’s behavior is highly variable, and therefore even a relatively large (in an everyday intuitive sense) number of observations might not accurately characterize the system.

While statistics teach us how to choose sample sizes to represent a population, certain assumptions about the underlying population (for example, the normality of a distribution) or the sampling process (such as the randomness) must hold to use these techniques. These assumptions do not necessarily hold for Internet measurements. Thus, often we are stuck between gathering too little data — which leaves us with a biased view — and expending a great deal of effort to gather a massive amount of data that “looks big enough” and therefore is seemingly beyond reproach.

The reality is that in both cases, we often have little understanding of a dataset’s representativeness. Small datasets might be perfectly fine in some cases, while seemingly massive datasets might be biased in some fashion. The worst part is that we often lack the tools or methodology to answer these “how much data is enough” questions.

Opacity
The Internet’s growth has also fueled ever-increasing complexity. This, in
turn, makes designing measurement experiments and interpreting results challenging.

Originally, the Internet simply forwarded packets from a source to a destination. This made measurement a relatively straightforward task. However, as we have introduced complexity into the forwarding of traffic — for example, in terms of proxies, firewalls, caches, replicas, NATs, ad injectors, and performance enhancers — an observation at one point might bear little resemblance to an observation of the same traffic at a different point. For instance, there is little about a data stream that a recipient can directly ascribe to the presumed source, because some of the data could have been altered in transit. Furthermore, various players on the Internet intentionally try to make the situation more opaque. For instance, applications camouflage themselves to avoid being blocked or throttled and encrypt communication to avoid external observation (whether malicious or for innocuous research purposes), while ISPs block Internet Control Message Protocol (ICMP) messages to avoid exposing their infrastructures to external scans.

Whether it rises from complexity or intentional obfuscation, the Internet's opacity makes the process of soundly measuring the system immensely difficult for two reasons. First, we always need increasingly clever methodologies to infer the network's true operation. Second, inevitably these methodologies are not simple and straightforward, so they raise the logistical burden of conducting measurements (by, for example, requiring many measurements to ascertain some particular behavior and ascribe it to some actor in the system). This more opaque Internet poses a huge challenge for the measurement and continued evolution of the system.

Ethics

With the crucial role of the Internet in our everyday lives, the ethical considerations of our work as Internet empiricalists again are becoming increasingly important. While well-managed but (potentially) disruptive experiments and measurements were acceptable in the past, the implications of disrupting peoples' communication have become far greater, and therefore now must receive heightened scrutiny.

As a simple example, sending a single probe to an arbitrary remote host is highly unlikely to be disruptive. On the other hand, the odds are good that transmitting probes to an arbitrary host at 1 Gbps for an hour will be viewed as a highly disruptive attack. Although the two ends of the spectrum are clear, where to draw the line between “non-disruptive” and “disruptive” is at best difficult.

Additionally, we use the Internet to exchange ever-more private information. Therefore, even passive measurement that does not perturb the system now must undergo increased scrutiny to ensure that any personal information captured is handled in an appropriate manner.

Another important aspect of measurement that requires ethical foresight is in terms of dealing with side effects. The Internet has dramatically democratized information exchange, and in many cases, freed information from government and traditional media control. However, this has triggered broad, state-sponsored surveillance efforts all over the world. In a non-trivial number of places, even seemingly benign communication across the Internet is viewed as incriminating. At the same time, some of our measurements can make traffic appear to be coming from a particular computer. Therefore, we must exercise care in not conducting measurements that will implicate individuals in activity that is viewed as problematic, but in which they have no part. Increasingly, researchers involved in Internet measurement must consider the non-technical side effects of their work.

In This Issue

This special issue attracted a large number of submissions. After several rounds of reviews and personal interactions with the authors, we selected five articles from 26 submissions. The selected articles provide a glimpse into diverse topics in this rich field of investigation.
Guest Editors’ Introduction

Alok Tongaonkar’s “A Look at the Mobile App Identification Landscape” provides a survey of methods that allow an ISP to understand which mobile applications generate certain traffic. ISPs need this information to monitor resource consumption by various applications, and to identify and block malicious activities. Yet assigning traffic to an application is challenging, because much of the traffic — regardless of the responsible application — runs over HTTPS (with encrypted payloads and common ports), and different applications might interact with overlapping sets of servers in the course of their operation.

“Measuring, Characterizing, and Avoiding Spam Traffic Costs” by Osvaldo Fonseca and his colleagues considers an interesting issue of which networks profit from, and which networks bear the cost of, delivering spam traffic through the Internet. The study measures the extent to which smaller networks bear the bulk of the cost of spam traffic delivery and sketches an algorithm that uses these measurements to identify profitable partnerships among networks for blocking spam.

Next, Glauber Gonçalves and his colleagues’ article “The Impact of Content Sharing on Cloud Storage Bandwidth Consumption” focuses on traffic exchanged between an organization and a cloud storage service such as Dropbox. By analyzing traces collected at several vantage points, this study quantifies the amount of potentially avoidable traffic due to repeated updates downloaded from the cloud, either by the device that already has these updates or by multiple devices sharing the content. The article consequently investigates the use of a shared cache to eliminate some of this traffic.

“Empirical Study of Router IPv6 Interface Address Distributions” by Justin Rohrer and his colleagues addresses the issue of IPv6 router topology mapping. While topology measurements through traceroutes are routinely performed across the IPv4 address space, the size of IPv6 address space presents hard challenges to conducting such measurements. The present study performs exhaustive probes of every /48 prefix within every advertised /32 address block and uses the resulting dataset to analyze subnetting and address usage practices by IPv6 network providers.

The final article in our collection — “Cuckoo Cache: A Technique to Improve Flow Monitoring Throughput” by Salvatore Pontarelli and Pedro Reviriego — is not a measurement study in itself, but rather addresses technology that enables large-scale measurements. Specifically, it proposes an enhancement to Cuckoo hashing, an efficient approach to implementing hash tables. An efficient hash table is key to a wide range of high-volume network measurements. In particular, this article demonstrates the benefits of their enhancement on the example of traffic flow monitoring on a link, where each packet leads to an update of a per-flow state, such as the amount of data carried by the flow.

We thank everyone for their submissions. We also thank the large number of colleagues who reviewed the submissions for this special issue. This issue would not have been possible without the reviewers’ time and expert opinions. We hope that IC’s readership will find these articles informative and enjoyable.

Michael Rabinovich is a professor in the Electrical Engineering and Computer Science Department at Case Western Reserve University. His research interests revolve around the Internet, especially concerning issues related to performance, measurement, and security. Rabinovich has a PhD in computer science from the University of Washington. He serves on the editorial boards of IEEE Internet Computing and ACM Transactions on the Web. Contact him at michael.rabinovich@cwru.edu.

Mark Allman is a senior scientist with the International Computer Science Institute (ICSI) and adjunct faculty in the Electrical Engineering and Computer Science Department at Case Western Reserve University. His current research focuses on network architecture, security, transport protocols, congestion control, and network measurement. Allman has an MS in computer science from Ohio University. He is a member of the ACM. Contact him at mallman@icir.org.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
A Look at the Mobile App Identification Landscape

The number of mobile devices and apps have grown tremendously in recent years, resulting in a dramatic increase in mobile traffic. This trend is expected to create a nearly 10-fold increase in global mobile data over the next 5 years, bringing mobile traffic analysis into focus. However, traditional traffic analysis approaches don’t work well for mobile traffic. Mobile apps possess unique characteristics that make it exceedingly difficult to determine which app generated a flow. Here, the author discusses the challenges in mobile traffic analysis and presents a survey of techniques that address these issues.

Recent years have seen a dramatic change in the way people access the Internet. The proliferation of mobile devices, such as smartphones and tablets, has altered the characteristics of network traffic. Typically, these mobile devices are used to access services over the Internet, using either a Web browser or through specialized mobile apps. According to Search Engine Watch,¹ there’s a clear trend of people spending an increasing amount of time on mobile devices, especially on mobile apps, as compared to desktops. They used data from comScore that measured the time that users spent on desktop and mobile devices (through mobile apps and browsers) for a year; the time remained constant at approximately 500,000 minutes per month for desktops from February 2013 to January 2014. The time spent on mobile browsers also remained more or less constant — around the 100,000-minute mark — in the same time period. However, the time spent by users on mobile apps increased significantly, from around 350,000 to more than 500,000 minutes. In fact, by January 2014 the time spent on mobile apps had increased to more than that spent on desktops, and this trend is expected to continue.

Another interesting trend is that the number of mobile users is increasing at a much faster rate than the number of desktop (or PC) users. According to a study published by Smart Insights,² in 2007 there were 400 million mobile users and 1,100 million desktop users. By 2014, the number of mobile users was nearly the same as desktop users (around 1,700 million). By 2015, the number of mobile users (1,900 million) exceeded the number of desktop users.
Measuring the Internet

(1,750 million). Further, Smart Insights also found that users spent 89 percent of time on apps versus 11 percent on browsers when using mobile devices.

This increased user interaction with mobile devices, and mobile apps in particular, has caused mobile traffic to have a greater share than desktop traffic. In fact, according to the Cisco Visual Networking Index Forecast, by the year 2019 global mobile traffic would have increased 10-fold. What this means is that there's a great need for tools and techniques that provide complete visibility into mobile traffic. ISPs want to identify the apps that cause the largest resource use. Moreover, network security operators need to know what app traffic is traversing the network to block potentially malicious activities. Thus, accurate app identification is critical from performance and security perspectives.

The App Identification Problem

We can pose the app identification problem as finding a relation that maps a network flow to the application that created the flow. More formally, if \( F = \{F_1, F_2, \ldots, F_n\} \) is the set of all observed flows in the network, and \( A = \{A_1, A_2, \ldots, A_m\} \) is the set of all the apps on a given mobile platform, then the problem is to find a \( RF(A) : F_i \rightarrow A_j \) such that \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \). Note that this is a many-to-one relation, because many flows can belong to the same app. We can address this problem with two subtly different coverage objectives in mind:

- **App coverage.** The aim is to identify all the apps running in the network — for example, if \( A_{all} \) represents the set of all apps running in the network, and \( A_{id} \) represents the set of all the apps identified from the traffic, then the objective is to maximize \( A_{id}/A_{all} \).
- **Flow coverage.** The aim is to identify the app for every flow in the network — for example, if \( F_{all} \) represents the set of all flows in the network, and \( F_{id} \) represents the set of all the flows that have been mapped to the originating app, then the objective is to maximize \( F_{id}/F_{all} \).

Various techniques for app identification target one or the other of these objectives. The distinction between the two objectives is as follows. The app coverage objective is realized as long as we correctly identify at least one flow generated by each mobile app running in the network.

Flow coverage, on the other hand, requires the originating app to be identified for each flow in the network. It’s easy to see that the app coverage objective is a special case of flow coverage, and in general, easier to achieve than flow coverage. The reason for making this distinction in the objectives is that different use cases for app identification have different requirements. For instance, an access-control system requires high flow coverage to prevent flows belonging to unwanted apps from entering a network. On the other hand, a forensic tool that needs to identify the apps in a network just requires high app coverage. Hence, the appropriate techniques that target a given objective can be chosen based on the use case requirement.

Challenges in App Identification

The traditional approaches to identifying applications or protocols don't work well for mobile traffic for a number of reasons. Port-based techniques don't work well for mobile traffic, as most of the traffic is carried over HTTP/HTTPS. Although machine-learning-based techniques — which use network behavioral features such as min/max/mean packet interval time or packet sizes — have been used successfully in the past for classifying network traffic, I couldn't find any evidence (in my own experiments or in the literature) that these techniques work as effectively for mobile traffic. (Behavioral-based techniques for mobile app classification could be an area of future research.)

Using the hostname of the servers contacted in HTTP flows also doesn’t work, because mobile apps typically contact a lot of servers belonging to different companies. For instance, an app such as Pandora or Netflix might contact the servers owned by the app developers (called origin servers) to get basic functionality such as authentication. The app also might contact content distribution networks (CDNs) for the actual content, such as songs or movies. Further, the app might contact third-party services such as Google Analytics and in-app advertisement providers such as DoubleClick or AdMob (both owned by Google). Moreover, many apps can contact the same servers. This has necessitated a need to come up with new techniques for identifying mobile apps in network traffic. Thus, in the following section I discuss how the landscape of mobile app identification techniques is evolving.
Another challenge for app identification is caused by a growing fraction of traffic being carried over HTTPS. The problem of dissecting encrypted traffic is a challenging one, even for desktop traffic. In some cases, such as the controlled environment in enterprises, this problem can be overcome using man-in-the-middle (MITM) solutions, which decrypt and re-encrypt traffic between the flows’ endpoints. In these solutions, given access to clear text using such MITM devices, the problem of app identification is identical for HTTPS and HTTP traffic. Hence, in the rest of this article, I don’t discuss encrypted traffic. Moreover, the percentage of HTTP traffic is still significant. This means that the challenges in mobile app identification in HTTP traffic needs to be overcome using new techniques.

Survey
Now that we have a sense of the problems and challenges obscuring adequate and accurate mobile traffic analysis, let’s look at how state-of-the-art techniques are attempting to address this problem.

Monitoring End Devices
The simplest technique for characterizing smartphone usage is to perform controlled experiments. Here a set of users is asked to use a device with certain apps installed and the usage behavior is monitored either on the device or in the network.

Hossein Falaki and his colleagues studied the mobile usage data from 43 users. They collected two datasets. The first one used Netlog on Windows Mobile (HTC Touch) and tcpdump on Android (HTC Dream) to record network traffic. The packet-level traces contained link-layer headers but there was no visibility into the mobile apps generating traffic in this dataset. The second dataset was collected on Android phones using a custom logging tool that provided an application-level view of smartphone traffic. Because of the difficulty of deploying continuous monitoring on a large number of end user devices, their data-collection methodology suffered from scaling issues. They acknowledge the lack of breadth in user population as a limitation of their work.

Xuetao Wei and his colleagues used a similar approach to profile mobile apps. They built a system called ProfileDroid, which modifies the Android platform to allow collection of diverse data — including network traffic — from Android apps. They access data collected from the users’ modified devices. However, their experiments also suffer from scaling issues and they restrict their evaluation to 10 runs of 19 apps by three users. Many research efforts — such as Meddle (www.meddle.mobi) and others — use the VPN APIs on the mobile platforms to get access to network traffic generated by an app. Thus, these systems are able to associate the app to the network flows. The advantage of techniques that try to identify app traffic on the device or redirecting the traffic is that they’re accurate — for example, they can determine exactly which app created the flow. However, as previously noted, this accuracy comes at the cost of performance/scaling issues.

User-Agent
Qiang Xu and his colleagues presented a large-scale study of mobile app characterization using one week of network traffic from an ISP. In contrast to previous works that used on-device monitoring, they used the user-agent field within the HTTP header to identify the apps. Mobile platform developers recommend putting app identifiers (a string or a number that uniquely identifies an app within an app marketplace such as Google Play or Apple App Store) in this field. However, this is not enforced by the platform vendors. In our study of over 100,000 Android and iOS apps, we saw that while many of the apps on iOS adhered to this, most of the ones on Android did not. Hence, this technique is not very useful when trying to obtain a high coverage in terms of the number of apps identified. Note that in the rest of the article I simply refer to the app marketplaces as “markets.”

Signature Generation
Shuaifu Dai and his colleagues proposed a signature-based technique, called NetworkProfiler, for identifying mobile apps. The signatures proposed by them have two components. First one is formed of the hostname that the traffic flows to/from. For instance, for Zedge — which is a popular app on Android for downloading wallpapers, ringtones, and notification sounds — the hostname component is *.zedge.net as Zedge flows contact different servers on zedge.net, such as fsa.zedge.net and fsh.zedge.net. The second component of the signature is a trie-like state machine on the method (GET/PUT/POST) and URL of the HTTP request. To generate
signatures, NetworkProfiler first runs an app multiple times and collects the network traces. Then the flow URL component is broken into different parts, such as the pathname and query string. The pathname is further broken into path-components and query into key-value pairs. Similar URLs are clustered together using hierarchical clustering and common patterns are extracted as so-called prefix tree acceptors (PTAs). These PTAs form state machines that are used as the second component of the signature. Figure 1 shows the URLs generated by the Zedge app in one run and Figure 2 shows the corresponding state machine based on these URLs. Given a new flow, its server hostname, HTTP method, and URL are matched against the set of all signatures, and the flow is associated with the app corresponding to the matching signature.

NetworkProfiler uses the monkeyrunner tool (https://developer.android.com/studio/test/monkeyrunner/) to automate the execution of apps. Because monkeyrunner randomly explores the app, it might not generate all the possible network flows. To overcome this, NetworkProfiler uses a custom dynamic analysis technique to achieve multipath execution of the app by using a seed execution. One of the challenges in this approach is that each app needs to be downloaded to produce a signature for it. Nicolas Viennot and his colleagues have developed a scalable infrastructure for automatically downloading Android apps from Google Play that addresses this problem.

**App Identifiers**

My colleagues and I used the findings from NetworkProfiler work to focus on app identifiers within advertising flows. The basic premise of this work is that many of the apps contain ads provided by various ad providers. These apps have an identifier that's used to identify the app to the ad provider, in order for the developer to be paid when an ad is viewed by a user through the app. Similarly, flows to analytics providers such as Google Analytics also contain identifiers.

This work focuses on ad flows and uses the app identifiers in ads to study smartphone usage behavior. In this technique, the network operator or a third party must sniff the manifest files of all the apps in a market and create the mapping from various app identifiers to the app from the package attribute. Then the operator or third party could match the app identifier from subsequent flows to the app using this mapping.

Figure 3 shows the manifest file for Zedge and the app identifiers used in ad flows for two different ad providers: AdMob and AdWhirl. Note that the identifier used by each ad provider might be different from the ones used by the other providers, and these identifiers could be the same or different than the app identifier in the market. The main shortcoming of using this technique is that even though the app coverage (the number of apps identified in the network traffic) is high, the flow coverage (the number of flows in network traffic labeled with the originating app name) is low because all non-ad flows are left unlabeled.

**Fingerprint Extraction**

To overcome the limitation of the app-identifier-based technique, Stanislav Miskovic and his colleagues proposed AppPrint, a system that uses the query parameters from HTTP URLs or from HTTP header strings to form app fingerprints. The idea of AppPrint is similar to NetworkProfiler. However, instead of forming comprehensive signatures that cover every behavior of the app, AppPrint aims to identify a few characteristics in the app flow that can be used for forming a fingerprint. The underlying intuition for their technique is to use parts of the HTTP URLs or strings from HTTP headers, called tokens, which are unique to the app as a fingerprint. To generate the fingerprints, AppPrint first collects network traffic by running the apps. The HTTP header information, including the query URL, is then tokenized using delimiters such as space, carriage return and line feed, and special characters such as “&” and “;” marks. Then AppPrint does a statistical analysis of each token to determine its prevalence. If a token is present only in the flows from a given app, then it can be used as a fingerprint for the app. This scheme's main drawback is that the fingerprint's quality depends on the training data. For instance, if a token appears to be unique to a given app from training data of some apps, it might still be present in other apps that haven’t been used for training. This can lead to false positives – for example,
incorrectly labeling a flow as belonging to an app when it doesn’t.

**Regression**

Qiang Xu and his colleagues built a system called Flow Recognition (FLOWR). The goal of FLOWR is to address the problem of AppPrint and NetworkProfiler; this requires generating flows for each app to derive fingerprints or signatures. In contrast, FLOWR tries to generate fingerprints from real network traffic by using some flows as seeds.

Typically, ad flows that contain identifiers that are same as the app identifier in a market are chosen as seeds. For instance, flows going to `googleads` contain “msid = X,” where X is the package name, such as `net.zedge.android`, which can also be used to identify the app in Google Play. FLOWR identifies the presence of an app using such fingerprints. Then, for each device (identified by its IP address or International Mobile Station Equipment Identifier) that originates this app flow, all of the flows occurring close by are grouped together in terms of time. At that point, features made up of the key-value pair tokens in the URL and hostname are extracted from these flow groups and compared across different devices. The intuition is that the flows occurring close to a given identified flow either belong to the same app or some other app running simultaneously. Because the probability of the other apps running on different devices being the same app is low, any feature that’s common across the different groups must belong to the app under consideration, as it’s known to be running on all devices.

The authors call this a regression technique, as they use information from well-known app flows to expand their knowledge and make predictions about other flows that are co-occurring. This way FLOWR can expand the fingerprint database by directly using network traffic and few well-known seeds.

**Rule Generation**

Self-Adaptive Mining of Persistent LEXical Snippets (SAMPLES) is a method that extracts rules that can be used to identify not just the app under consideration, but also other apps that haven’t been seen in training. This is in contrast to the aforementioned techniques, which analyze individual apps and try to extract
Measuring the Internet

either signatures or fingerprints to identify the
given apps.

At the core of SAMPLES is the idea that the
lexical context around an app identifier can be
used to form a rule that would extract the iden-
tifier from any flow with a matching context.
SAMPLES builds a repository of all possible
identifier strings for an app by parsing market
webpages as well as examining the metadata
files, such as the manifest file on Android, in
the app package. Then the app is executed in a
controlled environment and the resulting HTTP
flows are captured. The flows that contain
app identifiers are grouped together. Then the
lexical context is extracted from these flows.
Because many apps use the same third-party
libraries for development, the same lexical con-
text might be present in flows from many apps.

Rule 1 shows a sample rule constructed by
SAMPLES. It says that if a flow is destined for
the hostname (HST) googleads.g.doubleclick.net,
then extract from the URL parameters (PAR)
field everything after “msid=“ and check in the
Android app identifier repository whether it’s a
valid app identifier. If this check passes, the string
produced by the EXTRACT clause gives the app to
which this flow belongs. The last step is required
to avoid false positives. Note that creating the app
identifier repository is simple and can be done by
just crawling market webpages. It doesn’t require
downloading and execution of apps, which is the
real bottleneck for other techniques. Although
SAMPLES is the most systematic and general-
ized approach among all the state-of-the-art
techniques, it doesn’t do as well in terms of flow
coverage, as compared to app coverage.

App-Ident-Rule 1: IF HST: googleads.g.
doubleclick.net
Extract FROM PAR, msid=([nw.]+), AND Lookup
IN {Android app id}.

Search Engine

The Approximate Matching of Persistent LExi-
con using Search-Engines (AMPLES)\textsuperscript{17} method
addresses the problem of improving flow cover-
age for app identification by posing the problem
as an information-retrieval problem, where lexi-
cal similarity of short-text documents is used for
classification. Unlike some of the previously dis-
cussed works, which require execution of apps for
training, this system only performs lightweight
static analysis of app executable archives that are
commonly used to distribute apps through mar-
ketplaces. This is a big advantage, as the resources
required for collecting and analyzing app execut-
able archives are much less than actually install-
ing and executing apps. This system parses the
app executable archives to extract strings such as
app identifiers, key-value pairs, URLs, and URI
information that can help in identifying the app.
These strings are collected together into a docu-
ment that’s indexed using an off-the-shelf search
engine such as Apache Lucene (https://lucene.
apache.org). Thus, there is one document per app
in the marketplace.

When a flow is observed in the network,
its parsed into tokens such as hostname, key-
value pairs in query parameters, HTTP headers,
and URL path components using a deep packet
inspection (DPI) tool. These parsed tokens are
collected together to form a query that’s sent to
the search engine. The search engine provides a
matching score for this query for the documents
indexed by it, and which have a similarity score
above a certain threshold. Thus, for the flows for
which a match is returned by the search engine,
we can know the app that it belongs to based on
the document that’s returned, because each doc-
ument is labeled with a unique app identifier.

Note that the search engine could return mul-
tiple documents for a given flow. In this case, we
call the match a fuzzy match. This could happen
for a number of reasons, as follows. Many apps
use the same third-party libraries, and sometimes
flows from different apps using these third-party
services might have no distinguishing features.
Another reason for this multi-match could be that
the apps belong to the same family — for example,
they’re developed by the same developers, and
the developer reused the same code among mul-
tiple apps. AMPLES provides good flow coverage.
However, the quality of returned results depends
on how good the statically extracted features
are for identifying the app. This technique of
using a search engine to find a matching docu-
ment (or app) can be extended to use informa-
tion extracted from apps by executing them or by
static analysis of source code.

Mobile app identification is an important and
challenging problem with wide-ranging
applications. Already, many techniques have
been developed to address this problem, and
these techniques have improved greatly upon the chaotic situation of a few years ago. Even so, there's ample opportunity to continue to innovate in this area. Newer techniques can build on the lessons from existing ones to push the envelope further.

References


Alok Tongaonkar is a Data Scientist Director, leading the Center for Advanced Data Analytics (CADA) team at Symantec. His research interests include network security and management, especially optimizing performance. Tongaonkar has a PhD in computer science from Stony Brook University, New York. He’s a reviewer for many peer-reviewed publications, such as IEEE Transactions on Information Forensics Journal and IEEE Internet Computing. He’s a senior member of IEEE. Contact him at atongaonkar@acm.org.
Measuring the Internet

Measuring, Characterizing, and Avoiding Spam Traffic Costs

Spam messages propagate malware, disseminate phishing exploits, and advertise illegal products. They generate costs for users and network operators, but it’s difficult to measure the costs associated with spam traffic and determine who pays for it. The method presented here quantifies spam’s transit costs, identifying the routes traversed by spam messages. Combining spam traffic’s volume with traceroute measurements and a database of internetwork business relationships, the authors show that stub networks are subject to high spam traffic costs. An algorithm they present identifies networks that would benefit from cooperating to filter spam traffic at the origin.

Spam messages accounted for 90 percent of all email messages and generated approximately 216 Tbytes of traffic per day in 2013. The war against spammers is fought on multiple fronts. Recently, several proposals have focused on filtering spam at its origin, to prevent spam messages from reaching the destination and reduce network bandwidth consumption. However, in practice, spam is usually treated only at the destination email server, by filtering content just before it’s delivered to the end user. Although the volume of traffic created by spam might be small when compared with other sources, such as streaming video, spam is still an important problem for network administrators.

An autonomous system (AS) in the Internet is an entity registered with Internet resource allocation authorities. Each AS operates its own network, with end hosts, routers, and interconnecting links. To achieve global reachability, networks establish peering relationships to exchange traffic. Inter-AS peering relationships might be paid, such as when a regional AS buys transit from a global AS, or settlement-free, when two ASes agree to exchange traffic without a charge. Because of the nature of such peering relationships, sending and receiving spam messages could result in direct costs for ASes that pay for transit.

Here, we evaluate the cost of spam traffic at the granularity of individual ASes (for others’ work in this area, see

Osvaldo Fonseca, Elverton Fazzion, Italo Cunha, Pedro Henrique Bragoni Las-Casas, Dorgival Guedes, and Wagner Meira Jr.
Universidade Federal de Minas Gerais, Brazil

Cristine Hoepers, Klaus Steding-Jessen, and Marcelo H.P. Chaves
Brazilian Emergency Response Team and Brazilian Network Information Center
Measuring, Characterizing, and Avoiding Spam Traffic Costs

Related Work in Investigating Spam Traffic

Our data collection methodology draws heavily on previous work on topology measurements, converting traceroute measurements to autonomous system (AS)-paths, and identifying inter-AS business relationships. Our measurements, using Réseaux IP Européens (RIPE) Atlas, closely follow routes traversed by spam messages, avoiding errors due to commonplace asymmetric routing and rare violations of per-destination routing.

Researchers have studied the properties of spam traffic for more than a decade. More related to our work, researchers have studied the cost of the spam-sending infrastructure (servers and botnets) and how to filter spam at the origin AS. Researchers have also studied how to identify suspicious ASes in the Internet based on their networking practices. Our work is complementary and advances the state of the art by investigating the cost of spam traffic at the granularity of ASes, giving insight on incurred costs and how to mitigate it.

Overview

Our measurement approach lets us understand which ASes pay for and which ASes profit from spam traffic. Using five honeypots deployed in different countries, in three continents over 31 days, we observed the traffic generated by 133 million spam messages that were delivered to those honeypots. We measured 57,419 routes traversed by spam messages using traceroutes issued from Réseaux IP Européens (RIPE) Atlas daily. We mapped IP addresses observed in traces to the AS that originated the IP prefix and post-processed the resulting AS-level paths to remove Internet Exchange Points. Finally, we estimated traffic costs using Center for Applied Internet Data Analysis’s (CAIDA’s) database of AS relationships, which tells whether an AS pair has paid peering or settlement-free peering.

Our data shows that large, global ASes profit from spam traffic, as they exchange traffic with paying customers and with settlement-free peer ASes. Medium, regional ASes often lack a settlement-free peer AS to forward spam messages toward their destination, so at least some of the messages will be forwarded over links with paid relationships. Small, border ASes pay for the entirety of their spam traffic, as they rely on their providers for connectivity. Interestingly, ASes that originate large amounts of spam have more limited connectivity (less peering ASes) than ASes that receive spam, increasing overall spam traffic costs.

Finally, we propose an algorithm to identify pairs of ASes that would mutually benefit and save costs by filtering spam traffic close to its source. Our algorithm uses only information publicly available to ASes and could be executed by the ASes or as a service for them. Our methodology applies not only to spam, but also to other sources of unsolicited traffic, such as high-bandwidth distributed denial-of-service attacks.

Our evaluation shows that filtering can significantly reduce spam traffic costs, but only when an AS uses our algorithm to identify the few other ASes that could also benefit from such filtering, and therefore would be willing to act on such traffic. Our characterization indicates that global initiatives against spam might waste their efforts on ASes that aren’t ultimately interested in filtering spam traffic. Our

References

contributions are applicable today and might be useful in existing spam-filtering services.

Datasets and Methodology
To obtain our datasets, we collect spam messages, measure routes that spam messages traverse, identify ASes on each route, and infer spam traffic costs based on AS peering relationships. We now describe how we collect and combine our datasets. We report statistics and results on data collected between 8 September and 8 October 2015. Figure 1 illustrates our measurement infrastructure.

Spam Messages and Global Spam Traffic
We collect spam messages from five honeypots, machines that pass as email open relays and proxies. As honeypots are never publicly announced, we assume (and manual inspection indicates) that the only email messages they receive are from spammers that scan for email open relays and proxies. Our honeypots never forward the received spam messages, except for messages whose content indicates they’re test messages that spammers send to verify whether open relays and proxies work. Our five honeypots are hosted at educational and commercial networks in five different countries in Europe, North America, and South America.

We log all spam messages at each honeypot, and collect all messages to a central server daily. In the analyzed period, we collected 133 million spam messages from 56,051 IP addresses in 879 distinct ASes registered in 115 countries. We find most IP addresses (82.02 percent) sending spam use the honeypots as open relays and send few messages (26.75 percent of the total), behavior previously observed in stealthy botnets. The remaining IP addresses (17.98 percent) use the honeypots as proxies to send a large number of messages (73.25 percent of the total) — behavior consistent with that of dedicated spam servers.

Figure 1 shows a dedicated spam server in AS9 using our honeypot in AS5 as a proxy to send an email message to a recipient whose email domain is in AS10. The red curve shows the path that the spam message would take if it were forwarded by our honeypot.

Although our honeypots don’t forward spam messages, we do consider the traffic that would be generated if the messages were sent. This outgoing traffic would be 3.08 times larger than incoming traffic, because spam messages have recipients in multiple domains. Although our data amounts to a small fraction of spam traffic in the Internet and is collected at proxies and open relays, we believe our conclusions generalize to global spam traffic disseminated through these mechanisms. When reporting spam traffic volumes in the following section, we also parenthesize a rough estimate of what our observations would amount to when scaled to a global spam volume. In particular, we multiply the spam volume we observe by 6,250, the ratio between Symantec’s estimation of global spam volume and spam volume in our data. Our goal isn’t to provide accurate global traffic volumes, just approximate the order of magnitude of what ASes are likely to observe on the Internet. For better coverage of global spam traffic, our methodology can be applied to other spam datasets.

Measuring Internet Routes Traversed by Spam Messages
We estimate the routes traversed by spam messages with traceroute measurements. We issue traceroutes from RIPE Atlas vantage points in the same ASes as our honeypots, toward each destination domain observed in spam messages. We don’t issue traceroutes from the honeypots directly, to preserve their identity. RIPE Atlas is a distributed measurement platform with more than 8,000 vantage points and provides vantage points in all ASes that host our honeypots.

To measure routes from spammers to honeypots, we first identify the AS that hosts each IP address that sent spam to one of our honeypots. We then identify RIPE Atlas vantage points on these ASes and issue traceroutes from these vantage points. Again, to preserve the honeypots’ identities, we don’t issue traceroutes to the honeypots directly; we instead issue traceroutes to a RIPE Atlas vantage point (or its first reachable IP hop) in the same AS as the honeypot. RIPE Atlas provides good (40.05 percent) coverage of the 879 ASes from which we received spam messages, letting us measure routes for most (92.26 percent) spam messages that our honeypots receive. This avoids uncertainties due to violations of destination-based routing or asymmetric routes.

Considering the infrastructure depicted in Figure 1, we issue traceroutes from the RIPE Atlas vantage point in AS9 to the RIPE Atlas vantage point in AS7, then from the RIPE Atlas node in AS7 to the recipient’s email server. The red curve shows the measured routes.

Measuring the Internet
One limitation is that RIPE Atlas enforces strict rate limits on measurements, which prevents us from measuring all routes from spammers to our honeypots and from honeypots to all destinations. We operate with a budget of 500 traceroutes per honeypot per day. To maximize the usefulness of our measurement budget, we issue traceroutes to each honeypot from RIPE Atlas vantage points located in the 50 ASes that send the most spam to that honeypot and are covered by RIPE Atlas. From each honeypot, we also pick the 450 destinations that receive the most spam from that honeypot. We recompute the set of ASes and destination domains that send and receive the most spam daily, based on the spam we collected the previous day.

As the distributions of the number of spam messages from each AS and to each destination are heavily skewed, our daily budget still allows for expressive message coverage. Over the analyzed period, we can cover 92.25 percent of messages from spammers to honeypots on average, with insignificant variation because the set of spammers is stable over the one-month period. We can also cover 69.53 percent of messages from honeypots to destinations, with a standard deviation of 24.69 percent across different days as spam campaigns and the set of destination domains change throughout the month.

### Mapping Traceroutes to AS-Level Paths

We map IP-level traceroute measurements to AS-level paths using IP-to-AS mapping data from iPlane,6 which maps an IP prefix to the set of ASes that originate the prefix. If a router is unresponsive or an IP address isn’t mapped to any AS, but is surrounded by responsive routers with IP addresses that map to the same AS – for example, […, AS1, x, AS1, …] – we map the IP address to AS1.

If a router is unresponsive or an IP address isn’t mapped to any AS, but is surrounded by responsive routers with IP addresses that map to different ASes, such as […, AS1, x, AS2, …], we assume that traffic flows from AS1 to AS2. This happens on 4.59 percent of the paths. This assumption might impact the completeness of our results (when x is not in AS1 or AS2), but it never impacts the correctness of our results as data flows, even if indirectly, between AS1 and AS2. If a traceroute doesn’t reach the destination, we consider the route up to the last measured hop.

In the example shown in Figure 1, we convert the traceroute measurement from the RIPE Atlas vantage point in AS7 to the AS-path [AS7, IXP1, AS3, AS8, AS10] (IXP stands for Internet exchange point). If all routers in AS8 are unresponsive or have unmapped IP addresses, we obtain […, AS3, AS10] instead.

We use a Border Gateway Protocol (BGP) routing table dump obtained from one of the ASes that hosts one of the honeypots to verify the correctness of our mapping method. We compared the AS-paths obtained with the mapping process for routes from one of our honeypots with the (true) AS-paths in the BGP routing table used by the AS hosting the honeypot. We found that 89.16 percent of AS-paths from IP-to-AS mapping were identical to the BGP AS-paths, and that 99.39 percent had at most one different AS. Although IP-to-AS mapping might have errors, isolated wrong mappings don’t impact our ability to identify the overall flow of money.

### Computing Spam Traffic Costs

We identify which ASes pay for or profit from spam traffic using the CAIDA’s inter-AS peering relationship database.6,7 The CAIDA AS relationships database is known to have errors, but as far as we know, it’s the most accurate and complete database on AS relationships. CAIDA’s database classifies inter-AS peering relationships as either customer-to-provider or peer-to-peer. We assume customers pay providers to achieve global connectivity, and that peers exchange traffic free of charge. These assumptions are common in the literature.6

CAIDA’s database doesn’t list inter-AS relationships between transit ASes and IXPs. Reasons include that IXPs don’t provide transit themselves and that IXPs only provide connectivity between ASes. In this work, we’re interested in the peering relationship between ASes that exchange traffic at IXPs. We built a list of IP prefixes and AS numbers used by IXPs combining data from PeeringDB and previous work.6,10 We remove these IP prefixes from traceroutes and AS numbers from AS-paths. We also identified 10 peerings with content providers (Google, Amazon, and Microsoft) that weren’t present in CAIDA’s relationship database. These peerings appear in 2.30 percent of AS-level paths (8.14 percent of the messages). We manually labeled these
peerings as settlement-free, peer-to-peer relationships. With these modifications, CAIDA’s relationship database contains 97.85 percent of all AS relationships in our AS-level paths and can resolve all relationships in 93.73 percent of AS-level paths. Our modifications let us compute all AS relationships traversed by 99.33 percent of the messages for which we measured routes.

In the example topology in Figure 1, we place providers above customers and show provider-to-customer relationships with solid lines. We show settlement-free peering relationships with dashed horizontal lines. We remove IXP1 from the AS-path before computing traffic costs. Finally, payments would flow from AS9 to AS7, and from AS10 to AS8, to AS3. We show content provider networks peering with Verizon and AS4.

Spam Traffic Costs in the Internet

Now we present our findings on spam traffic costs and provide representative examples.

We estimate spam traffic costs for each AS. As peering contracts between different ASes are private, we can’t know how much transferring each byte costs each AS. We’re conservative and make no assumptions on the cost of traffic. We instead estimate spam traffic costs as the spam traffic volume exchanged with providers and customers.

ASes profit from spam traffic exchanged with customers, and pay for spam traffic exchanged with providers. We study the net spam traffic for each AS, which is the volume of traffic exchanged with customers minus the volume of traffic exchanged with providers. A positive net spam traffic means the AS profits from spam traffic, and vice-versa. Although we find spam traffic is often unbalanced across settlement-free peering links, we consider that traffic on these links doesn’t incur costs and ignore it when computing net spam traffic.

Figure 2 shows the median and quartiles (boxes), the 5th and 95th percentiles (dashed lines), as well as outliers of net spam traffic for ASes in our data. ASes with positive net spam traffic are in the “profit” column, and vice versa. Note the logarithmic scale on the y axis.

We classify ASes by their customer cone sizes, meaning the number of other ASes they can reach without using a provider (that is, free of charge). We find most ASes pay for or profit from a small volume of spam traffic. To focus on ASes more significantly impacted by spam traffic, Figure 2 doesn’t include ASes with an average net spam traffic rate between ±16 Kbps (±100 Mbps after multiplying by 6,250 to scale to global spam volumes). Large ASes, with customer cones including more than 100 ASes, rarely pay for and often profit from spam traffic. This is because
these ASes profit from exchanging spam traffic with their customers, and forward spam traffic through peering ASes free of charge. Level 3 (AS3356) transits 28.17 percent of the spam messages in our data, and profits from a net spam traffic rate of 2.77 Mbps (17.36 Gbps). Notice that large ASes profit twice from spam traffic whenever they receive a spam message from a customer and forward it to another customer.

Medium ASes, with customer cones including tens of ASes, rarely profit from spam traffic and often pay for spam traffic. Figure 2 shows one medium AS with a profit, and all other medium ASes with losses. Medium ASes pay for spam traffic in two situations:

1. Medium ASes pay a provider for outgoing spam traffic whenever they can’t forward it to peering ASes free of charge. This amounts to 49.37 percent of total spam traffic costs, on average, for medium ASes in our data. We find forwarding traffic through peering ASes free of charge reduces spam costs for traffic by 34.68 percent, on average, for ASes in our data.

2. Medium ASes pay for incoming spam traffic received from their providers. This accounts for the remaining 50.73 percent of total costs for medium ASes in our data. We note that ASes that originate spam have worse connectivity than ASes that receive spam (their average number of provider-plus-peer ASes is 1.38 and 4.31, respectively). This causes spam traffic to go high in the Internet’s transit hierarchy and incurs costs on all ASes on the downstream portion of the path.

Medium ASes profit from exchanging spam traffic with their customers. However, the previously discussed costs might result in negative net spam traffic when medium ASes originate or receive spam, as originated and received spam traffic isn’t exchanged with any customer and doesn’t generate revenue.

The only medium AS we observe with a profit is Hinet (AS9680, a customer cone with 24 ASes), which profits from receiving an average 689 Kbps (4.3 Gbps) of outgoing spam traffic from its customers and forwards 80.44 percent of it through peering ASes. Further investigation shows all of Hinet’s spam comes from its Taiwan-based subsidiary (AS3462). Hinet has been reported before as a spam haven, and our results explains why it remains so: Hinet can charge spammers and forward most spam for free.

Small and stub ASes, with customer cones smaller than 10 ASes, pay for the bulk of spam traffic costs. These ASes originate and receive spam traffic, incurring losses from exchanging spam traffic with their providers. Although the three outliers are ASes hosting our honeypots (with customer cone sizes of 1, 3, and 9), we observe that 25 percent of ASes pay for spam traffic rates higher than 237 Kbps (1.48 Gbps).

Filtering Spam Traffic

Next, we use our findings to propose an algorithm to identify ASes interested in filtering spam traffic. Because some ASes profit from spam traffic, they wouldn’t be interested in spending human and computational resources to filter spam traffic. Even ASes that pay for spam traffic wouldn’t be interested in filtering spam traffic on the downstream path — that is, when they would forward spam traffic to one of their clients. For example, in Figure 1, AS7 would be willing to filter spam coming from AS9 and destined to AS10, on behalf of AS10, because it would receive payment for this traffic from AS9 but avoid its own payment to AS3. However, AS7 would be unwilling to filter spam.
Measuring the Internet

Algorithm 1. Choosing partners for filtering spam. Filtering all traffic from a source (spammer) to a destination (SMTP server) is possible at the packet level based on IP addresses.

```plaintext
Input: Spam messages, AS-level route measurements, inter-AS business relationships
Output: C[x][y]: amount of traffic x and y can filter to reduce spam traffic costs
1: for each AS x do
2:    S_in ← set of spam messages destined to x received from providers
3:    for each AS y on the paths traversed by messages in S_in do
4:        if y reaches x through a provider then
5:            C[x][y] ← volume of spam traffic between x and y
6:    S_out ← set of spam messages through x sent to providers
7:    for each AS y hosting destination domains for messages in S_out do
8:        if y receives messages through x from a provider then
9:            C[x][y] ← C[x][y] + volume of spam traffic between x and y
```

Figure 3. Traffic cost savings when filtering spam close to the source. We show a line for the scenario where each AS cooperates with all other ASes, where each AS cooperates with 10 other ASes following our algorithm, and where each AS cooperates with 15 other ASes chosen at random.

From AS10 to AS9 on behalf of AS9 (after having already incurred the payment to AS3 for delivering this traffic) for less than the profit from delivering this traffic to AS10. A practical solution would need to filter spam traffic at or close to its originating AS.

Algorithm 1 summarizes our proposal for identifying possible spam traffic-filtering partnerships. Each AS x in the Internet can collect spam messages it receives from a provider (that is, it pays for). Then AS x uses our methodology to identify which other ASes pay to send x’s spam messages to a provider and would profit from filtering these messages. Next, AS x contacts these ASes in an attempt to establish spam traffic-filtering agreements. Similarly, each AS x can collect spam messages it forwards to a provider, then use our methodology to identify which ASes hosting the destination domains pay to receive the messages. At that point, AS x can contact these ASes and offer to filter spam traffic destined to them. Filtering all traffic from a source (spammer) to a destination (SMTP server) is possible at the packet level based on IP addresses. Selective filtering of spam messages among legitimate messages requires more complex (for example, stateful) processing at the intermediate AS.

The solid blue curve in Figure 3 shows the distribution of potential savings in spam traffic costs for all medium and small ASes in our data. We compute potential savings for each AS as the fraction of spam traffic exchanged with providers that can be filtered. More precisely, we compute potential savings for an AS x as the ratio of x’s spam traffic costs when all ASes agree to filter all messages that incur costs by x’s total spam traffic costs without filtering. The curve shows that filtering, even when driven by profit, can mitigate all spam traffic costs for 60.14 percent of ASes and reduce spam traffic costs by at least half for 88.66 percent of ASes.

The red dashed curve in Figure 3 shows the distribution of spam traffic cost savings when each AS establishes filtering agreements with 10 ASes identified by our algorithm. Each AS establishes agreements with the 10 ASes that lead to the highest reduction in spam traffic costs, as computed by our methodology. After choosing an AS to establish an agreement, we recompute spam traffic costs — that is, we re-run Algorithm 1, before picking the next. We observed that
cooperating with as few as 10 ASes is enough to achieve significant savings when ASes are chosen intelligently. Cooperating with the top 10 ASes mitigates all spam traffic costs for 37 percent of ASes and reduces spam traffic costs by at least half for 77.06 percent of ASes.

Finally, the dotted black curve close to the y axis in Figure 3 shows the distribution of expected savings when AS x establishes filtering agreements with 15 ASes at random. We approximate expected savings as the average savings of more than 100 random samples of 15 filtering agreements. The curve shows that picking partnerships at random doesn’t allow for significant savings. This happens because most ASes (90.28 percent) either forward no spam for x or are uninterested in filtering x’s spam traffic.

In using our measurement methodology and data to characterize the cost of spam traffic in the Internet today; we found that large ASes profit from spam traffic while medium and small ASes pay. We used this insight to propose an algorithm to identify pairs of ASes that might benefit from cooperating to filter spam traffic close to the sources. Our algorithm uses only data that’s readily available and could be run by any AS or be provided as a service. Our evaluation shows that our algorithm can help ASes find other ASes that could benefit and therefore might be willing to cooperate in filtering spam. Results show that it performs significantly better than choosing ASes to cooperate with at random. Because our contributions are applicable for current spam-filtering efforts, they could ultimately lead to reduced spam traffic on the Internet.

Acknowledgments
This work was partially funded by NIC.Br, Fapemig, CNPq, CAPES, and by projects InWeb [MCT/CNPq 573871/2008-6], MASWeb [FAPEMIG-PRONEX APQ-01400-14], and EU-Bra-BIGSEA (H2020-EU.2.1.1 690116, Brazil/MCTI/RNP 6), MASWeb (FAPEMIG-PRONEX APQ-01400-14), and CAPES, and by projects InWeb (MCT/CNPq 573871/2008-6), MASWeb [FAPEMIG-PRONEX APQ-01400-14], and EU-Bra-BIGSEA (H2020-EU.2.1.1 690116, Brazil/MCTI/RNP 6). This work was partially funded by NIC.Br, Fapemig, CNPq, CAPES, and by projects InWeb [MCT/CNPq 573871/2008-6], MASWeb [FAPEMIG-PRONEX APQ-01400-14], and EU-Bra-BIGSEA (H2020-EU.2.1.1 690116, Brazil/MCTI/RNP 6).

References

Osvaldo Fonseca is a PhD student in the Computer Science Department at the Universidade Federal de Minas Gerais (UFMG), Brazil. His research interests include computer networks and data mining, with an emphasis on the development of techniques to combat spam at its source. Fonseca has an MS in computer science from UFMG. Contact him at osvaldo.morais@dcc.ufmg.br.

Elverton Fazzion is a master’s student in the Computer Science Department at the UFMG. His research interests include computer networks, with an emphasis on Internet measurements, and in the development of techniques to combat spam at its source. Fazzion has a BS (with honors) in computer science from UFMG. Contact him at elverton@dcc.ufmg.br.

İtalo Cunha is an assistant professor at the Computer Science Department at UFMG. His research interests include improving network performance and reliability, designing new mechanisms for network monitoring, modeling Internet routing, and supporting novel network applications. Cunha has a PhD in computer science from Université Pierre et Marie Curie–Sorbonne Universités. Contact him at cunha@dcc.ufmg.br.

Pedro Henrique Bragoni Las-Casas is a PhD student in the Computer Science Department at UFMG. His research...
interests include distributed systems, machine learning, data mining, and computer networks. Las-Casas has an MS in computer science from UFMG. Contact him at pedro.lascasas@dcc.ufmg.br.

Dorgival Guedes is an associate professor in the Computer Science Department at UFMG. His research interests include computer networks, distributed systems, and operating systems — especially where they relate to the scalability of globally distributed applications — focusing on areas such as cloud computing, Big Data, and software-defined networks (as well as network security issues, such as spam). Guedes has a PhD in computer science from the University of Arizona, Tucson. Contact him at dorgival@dcc.ufmg.br.

Wagner Meira Jr. is a full professor in the Computer Science Department at UFMG. His research interests include scalability and efficiency of large-scale parallel and distributed systems (from massively parallel to Internet-based platforms), and data mining algorithms, their parallelization, and application to areas such as cybersecurity, information retrieval, bioinformatics, and e-governance. Meira has a PhD in computer science from the University of Rochester. Contact him at meira@dcc.ufmg.br.

Cristine Hoepers is the general manager of the Brazilian Emergency Response Team and Brazilian Network Information Center (CERT.br/NIC.br). Her research interests include the development of Internet security best practices and end-user awareness materials, and the coordination of the honeyTARG Project (a Chapter of the Global Honeynet Project). Hoepers has a PhD in applied computing from the Brazilian National Institute for Space Research (INPE). Contact her at cristine@cert.br.

Klaus Steding-Jessen is the technical manager of CERT.br/NIC.br. His research interests include infrastructure, training and trend analysis (the latest as part of the honeyTARG Project). Steding-Jessen has a PhD in applied computing from INPE. Contact him at jessen@cert.br.

Marcelo H.P. Chaves is a senior security projects analyst at CERT.br/NIC.br. His research interests include infrastructure, research and development, and trend analysis (the latest as part of the honeyTARG Project). Chaves has an MS in applied computing from INPE. Contact him at mhp@cert.br.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
PURPOSE: The IEEE Computer Society is the world’s largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine Computer, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEBSITE: www.computer.org

OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

Computer: The flagship publication of the IEEE Computer Society, Computer, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals: The society publishes 13 magazines, 19 transactions, and one letters. Refer to membership application or request information as noted above.

Conference Proceedings & Books: Conference Publishing Services publishes more than 175 titles every year.

Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.

Technical Committees: TCs provide professional interaction in more than 45 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

NEXT BOARD MEETING

13–14 November 2016, New Brunswick, NJ, USA

EXECUTIVE COMMITTEE

President: Roger U. Fujii
President-Elect: Jean-Luc Gaudiot; Past President: Thomas M. Conte;
Secretary: Gregory T. Byrd; Treasurer: Forrest Shull; VP, Member & Geographic Activities: Nita K. Patel; VP, Publications: David S. Ebert;
VP, Professional & Educational Activities: Andy T. Chen; VP, Standards Activities: Mark Paulik; VP, Technical & Conference Activities: Hausi A. Müller; 2016 IEEE Director & Delegate Division VIII: John W. Walz; 2016 IEEE Director & Delegate Division V: Harold Javid; 2017 IEEE Director-Elect & Delegate Division V: Dejan S. Milojčić

BOARD OF GOVERNORS

Term Expiring 2016: David A. Bader, Pierre Bourque, Dennis J. Frailey, Jill I. Gostin, Atsunori Goto, Rob Reilly, Christina M. Schober

Term Expiring 2017: David Lomet, Ming C. Lin, Gregory T. Byrd, Alfredo Benzo, Forrest Shull, Fabrizio Lombardi, Hausi A. Müller

Term Expiring 2018: Ann DeMarle, Fred Douglass, Vladimir Getov, Bruce M. McMillin, Cecilia Metra, Kunio Uchiyama, Stefano Zanero

EXECUTIVE STAFF

Executive Director: Angela R. Burgess
Director, Governance & Associate Executive Director: Anne Marie Kelly
Director, Finance & Accounting: Sunny Hwang
Director, Information Technology & Services: Sumit Kacker
Director, Membership Development: Eric Berkowitz
Director, Products & Services: Evan M. Butterfield
Director, Sales & Marketing: Chris Jensen

COMPUTER SOCIETY OFFICES

Phone: +1 202 371 0101 • Fax: +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720
Phone: +1 714 821 8380
Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS
Phone: +1 800 272 6657 • Fax: +1 714 821 4641 • Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • Fax: +81 3 3408 3553
Email: tokyo.ofc@computer.org

IEEE BOARD OF DIRECTORS

President & CEO: Barry L. Shoop
President-Elect: Karen Bartleson
Past President: Howard E. Michel
Secretary: Parviz Famouri
Treasurer: Jerry L. Hudgins
Director & President, IEEE-USA: Peter Alan Eckstein
Director & President, Standards Association: Bruce P. Kraemer
Director & VP, Educational Activities: S.K. Ramesh
Director & VP, Membership and Geographic Activities: Wai-Choon (Lawrence) Wong
Director & VP, Publication Services and Products: Sheila Hemami
Director & VP, Technical Activities: Jose M.F. Moura
Director & Delegate Division V: Harold Javid
Director & Delegate Division VIII: John W. Walz

revised 10 June 2016
The Impact of Content Sharing on Cloud Storage Bandwidth Consumption

Content sharing in cloud storage leads to multiple downloads of the same content when users synchronize devices. These downloads contribute to bandwidth waste and increase server workloads. Here, the authors investigate traffic generated by Dropbox and use data collected from four networks to show that a large fraction (57–70 percent) of downloads generated by Dropbox users is associated with content shared among multiple devices. They present an alternative synchronization architecture that uses caches to offload storage servers from such downloads. Their experiments show that the approach cost-effectively avoids most repetitive downloads, benefiting service providers, the network, and end users.

Cloud storage is currently one of the most popular Internet services, generating traffic volume that has been increasing at a fast pace. Indeed, the entrance of big companies (such as Google, Microsoft, and Apple) into this market confirms the lively scenario. Dropbox, a leader in the cloud storage market, has surpassed the mark of 400 million users, uploading 1.2 billion files to the Internet every 24 hours in 2015. Such services offer a practical and safe environment for both domestic and enterprise users to store and share data, facilitating content organization and collaborative work. Yet, popular features of these services—notably content sharing—pose an extra load for servers and the network, as data shared among multiple user devices might require several transfers from remote servers. This holds even if devices are close to each other (for example, within a campus network), and despite Dropbox’s efforts to implement device-to-device synchronization with the LAN Sync protocol. Such downloads ultimately waste network bandwidth and increase the workload at the cloud servers.

Cloud storage services employ mechanisms to reduce network traffic, such as compression and deduplication. Although each of these mechanisms reduces the traffic between the cloud and user devices by up to 24 percent, they don’t target content sharing and, as such, have limited effect on downloads of the same content to synchronize multiple devices. This is...
The Impact of Content Sharing on Cloud Storage Bandwidth Consumption

worrisome given previous observations that downloads account for higher traffic than uploads in cloud storage.\(^\text{1,6}\)

At the same time, Internet traffic in general presents significant redundancy\(^\text{2}\) caused by, for example, the download of a single content by multiple users. Web caching and content delivery networks are classical solutions to offload servers and remove cross-border traffic from the network. Intuitively, these solutions could be applicable to cloud storage as well. However, most cloud storage providers don’t implement any distributed synchronization architectures,\(^\text{4}\) such as the deployment of caches nearby to end users who are connected far from datacenters. Is the traffic caused by content sharing significant for providers and, if so, are content-sharing characteristics such that caching could cost-effectively reduce its impact on servers?

In this article, we address to what extent content sharing in cloud storage leads to repetitive downloads from the cloud (and thus bandwidth waste) in current networks. This problem has only been discussed to date, though still in a preliminary manner, in our prior work.\(^\text{8}\) Here, we perform a thorough investigation of this issue. We characterize content sharing among Dropbox users relying on traffic traces collected in four distinct networks: two university networks (one in South America and one in Europe) and two points of presence (PoP) of a European ISP.

We also assess the impact of content sharing on cloud storage traffic by quantifying the downloads associated with the same content shared by multiple devices within the monitored networks. We find that a large fraction (57–70 percent) of the downloads from Dropbox servers falls into this category. Moreover, a significant fraction of such traffic (up to 25 percent of Dropbox related incoming traffic in the monitored networks) is likely caused by the download of content replicas and, therefore, is avoidable.

We then explore whether the introduction of network caches would reduce the number of avoidable downloads in a cost-effective way. We propose a modification to the synchronization architecture of Dropbox, introducing caches to temporarily hold user updates. We evaluate this architecture in various setups, considering both typical scenarios based on our datasets and simulations with larger user populations and/or content sharing.

We show that even a reasonably small cache (for example, 70 Gbytes) could offload servers from handling almost all avoidable downloads. Even more, our proposed architecture is cost-effective: taking the traffic observed in one of our datasets, we find that 92 percent of the costs for serving avoidable downloads can be recovered after discounting the costs of the cache. Moreover, such benefits tend to increase, reaching 95–97 percent if we consider adequate cache sizes (for example, 280–500 Gbytes) and scenarios where the cache is deployed to cover larger user populations (for example, at large ISPs) and/or user populations sharing more content.

Overall, our results show that storage providers have incentives to deploy the caching-based architecture, which additionally would remove cross-border traffic from the Internet and reduce synchronization time for end users.

**Content Synchronization in Dropbox**

To begin, let’s look at how Dropbox synchronizes content.

**Basic Mechanisms**

Dropbox synchronizes content by relying on two main concepts: devices and namespaces. Users can register several devices in the system. During this process, users select an initial folder from which files are synchronized with the cloud. This initial folder is visible from any other device belonging to the user. Users might share content with other users by creating shared folders, which are visible in all devices of all users participating in the sharing. Both initial folders and shared folders are the root of independent directory trees, where actual files are stored, and are called **namespaces** in the Dropbox system.\(^\text{9}\)

We focus on the Dropbox desktop client, because it’s responsible for more than 75 percent of the Dropbox traffic in 2014.\(^\text{1}\) Devices using this client usually keep a local copy of all files present in the user’s namespaces. The addition of any content in a namespace triggers content propagation: all devices having the Dropbox desktop client and registering the namespace retrieve the content immediately if online, or as soon as they come back online.

Dropbox controls the status of namespaces by means of a notification protocol, which wasn’t encrypted until mid-2014. Each namespace is associated with a journal identifier (JID), representing its latest version. Devices discover when namespaces are outdated by periodically exchanging a list of namespaces and respective
JIDs with Dropbox servers. If any namespace is outdated, the device executes several transactions with Dropbox servers until all namespaces become synchronized with the cloud.

We define an update as the steps that a device needs to take to move a namespace from a JID to its next JID value. Updates include files that have been added to the namespace and all metadata and commands that manipulate the namespace, such as to delete files and create folders.

By observing messages of the notification protocol, it’s possible to identify when namespaces are updated. Indeed, we developed a methodology in previous work to collect a long-term dataset about Dropbox namespaces, which includes the traffic volume exchanged in each JID transition of a large sample of namespaces — that is, an estimation of the update sizes.

**Synchronization Architecture with LAN Sync**

Dropbox deploys the LAN Sync protocol for synchronizing devices in LANs, which might prevent downloads from the cloud. The protocol works as follows: First, devices periodically broadcast information about their namespaces. Any device in the LAN can form a list of possible neighbors for future synchronizations. Next, an outdated device checks the status of neighbors before retrieving updates from the cloud. Device-to-device synchronization takes place if the namespace is already updated in any device within the LAN.

Dropbox broadcast messages reach a limited number of devices. Moreover, devices sharing namespaces must be online simultaneously to allow the LAN Sync protocol to work effectively.

In Figure 1, some updates (solid red arrows) need to be retrieved from the cloud to synchronize devices: update generated (a) inside the network (Dev1 is offline) and (b) outside the network (Dev2 is offline, Dev1 is back online and retrieves upd1 and upd2 from the cloud). Our proposal for an alternative synchronization architecture: update generated (c) inside the network (Dev1 is offline) and (d) outside the network (Dev2 is offline, Dev1 is back online and retrieves upd1 and upd2 from the cache).
devices, even if the same updates have already been observed in the network — for example, when LAN Sync isn’t effective. We call those cases avoidable downloads. Avoidable downloads occur either because the update has been previously uploaded by a device in the network, or because multiple devices download a single update generated elsewhere.

**Characterizing Avoidable Downloads**

Next, we closely analyze data we captured, to better understand the effects and dimensions of downloads that are avoidable.

**Datasets and Methodology**

We rely on data captured and prepared in our previous work, where we modeled the workload of cloud storage. Different from that work, here we analyze content sharing and assess its impact on storage traffic. We collected the datasets by monitoring Dropbox traffic at four vantage points. Campus-1 and Campus-2 are distinct campus networks in South America and Europe. Campus-1 has a user population of ≈57,000 people, whereas Campus-2 serves ≈15,000 people. PoP-1 and PoP-2 monitor customers at PoPs of a European ISP, aggregating ≈25,000 and ≈5,000 households, respectively.

Our data includes flow-level information containing the volume exchanged by clients with Dropbox servers, and metadata extracted from Dropbox notification messages. We collected the latter by means of deep packet inspection and include, for each notification, the device ID and the list of namespaces with respective device JIDs. Note that client IP addresses are anonymized and notifications offer no hints about users’ identities. In total, we observed around 23 Tbytes of Dropbox traffic, 27,428 unique Dropbox clients exchange notifications with Dropbox devices, and 61,419 unique namespaces.

Dropbox clients exchange notifications with servers once a minute when online. When a namespace is updated, additional Dropbox traffic appears in the network, and a notification message announcing the new JID is sent out by the client. We thus process traces (see also our previous work[9]) to estimate each update’s size, correlating flow volumes with notification messages, as follows:

- flows and notifications are first grouped per client IP address;
- flows and notifications overlapping in time form a synchronization cycle;
- synchronization cycles are classified as upload, download, or mixed, based on downstream and upstream volumes; and
- update sizes are estimated for clean cycles, while the remaining cycles are discarded.

We consider a synchronization cycle to be clean when a single device is active — either uploading or downloading — given an IP address, or if multiple devices are active sharing an IP address, but a single namespace is changed. We assume the latter to be a typical network address translator (NAT) scenario, where one device uploads content that’s spread to peer devices. We know which devices are active even behind NATs, and the likely uploader, thanks to the device IDs found in notification messages. Among situations that prevent us from estimating update sizes, we highlight multiple devices editing various namespaces simultaneously behind a NAT; and the synchronization happening when devices reappear online, which often involves many namespaces and both uploads and downloads. Finally, the volume in a clean cycle is divided equally among updates in the cycle, for simplicity.

Overall, we retain more than 63 percent of the Dropbox traffic for our analysis (see Table 1). We use the resulting dataset to study avoidable downloads. We track all updates of namespaces and mark downloads as avoidable when a namespace is updated to a specific JID in a device, and at least one other device registering the namespace has announced the same JID. Note that our methodology doesn’t take into account synchronizations performed using LAN Sync, because such traffic doesn’t reach our probes. We thus identify avoidable downloads after LAN Sync actuates.

**Results**

Table 1 lists results, including the number of namespaces shared by at least two devices in the monitored networks (that is, shared namespaces); traffic volumes retained for the analysis; traffic volumes associated with shared namespaces, and avoidable downloads; and datasets’ duration.

**Shared namespaces.** Table 1 shows that the percentage of shared namespaces in campuses (≥33 percent) is higher than in PoPs (≥26 percent). The number of distinct devices registering each shared namespace is, however, small. More than 90 percent of the shared namespaces
are registered by two devices only. The fraction of namespaces shared by at least three devices is somewhat higher in the campus datasets, reaching 7–9 percent. In the PoP datasets, this fraction falls to the 4–6 percent range. Such a difference is unsurprising and has been associated with the use of Dropbox for collaborative work in campuses, and the synchronization of a user’s different devices at home.10

We find that users’ interest on namespaces tends to last for a short time: after an initial period, the number of accesses (that is, updates and downloads) in a namespace becomes negligible. We illustrate this behavior in Figure 2a by showing cumulative distribution functions of the number of days between the first and last access in the first 60 days of life of namespaces. We refer to this time period as the namespace lifespan. Moreover, we focus on namespaces created during our captures and consider only namespaces seen online for at least 60 days.

Notice how all accesses occur on the day the namespace first appears for 22–27 percent of the namespaces. The median lifespan of a namespace is around one month. There is, however, a non-negligible number of namespaces with long lifespans: for example, around 20 percent of the namespaces in PoP-1 still present activity 50 days after the first access. Yet, overall, we find that shared namespaces tend to have short lifespans—that is, their accesses occur with a strong temporal locality, which seems to favor the deployment of network caches.

Avoidable traffic. Table 1 shows a high volume of downloads in all datasets. We observe that 44–68 percent of the traffic corresponds to downloads, and 57–70 percent of such download traffic belongs to shared namespaces. This raises
The Impact of Content Sharing on Cloud Storage Bandwidth Consumption

a question about how much traffic is associated with content replication. Not all downloads are avoidable, because devices might retrieve content uploaded in other networks, for example, when a user has remote devices, or namespaces shared with users located somewhere else. We see, however, that the percentage of avoidable downloads is quite significant. Overall, we find that up to 25 percent of the Dropbox download traffic happens to synchronize content that has been observed in the same network.

Figure 2b presents the total download volume that’s avoidable per update. Note that most updates that cause such downloads generate little traffic. Yet, a non-negligible portion results in large volumes. For example, while 25 percent of the updates in Campus-1 generate at least 100 Kbytes of replication, more than 3 percent of the updates surpass 1 Mbyte, reaching up to 1 Gbyte of replication.

Summary. Our analysis confirms that downloads dominate Dropbox traffic, and a significant part of such downloads is avoidable, contributing to increased costs for the provider. Table 1 shows significant percentages of avoidable downloads in four networks, each with a different prevalence of NATs, indicating that results aren’t influenced by the data discarded by our methodology to estimate the size of updates.

Shared namespaces have a limited lifespan. Updates on those namespaces are typically small and present strong temporal locality. These results motivate us to investigate next whether the introduction of network caches could reduce the number of avoidable downloads in a cost-effective way.

A New Synchronization Architecture

We propose a new synchronization architecture for cloud storage that consists of introducing network caches to temporarily hold user updates in the network. We aim to enable device synchronization, without the need to retrieve content from the cloud in scenarios where the LAN Sync protocol is ineffective. Figures 1c and 1d illustrate our proposal in the same cases shown in Figures 1a and 1b. A storage cache is installed to cover several networks and possibly thousands of customers — for example, multiple LANs in a large campus, complete ISP networks, or multiple PoPs. As a consequence, devices can potentially find updates without retrieving content from the cloud.

Synchronizing devices using the storage caches works as follows. The discovery of caches is orchestrated by Dropbox protocols when devices log in to the system. Dropbox servers inform clients about the closest cache in the network. Devices always send updates in namespaces to the closest cache (see Figure 1c, step 1). The cache stores updates locally, also forwarding them to the cloud. As soon as devices receive notifications about updates, they contact the closest cache. If the pending updates exist in the cache (that is, a cache hit), the cache delivers the content directly to devices. Otherwise (that is, a cache miss), the cache retrieves the content from the cloud and forwards it to the requesting devices (see Figure 1d, steps 2 and 3). After any request from a client, the cache executes internal replacement policies to guarantee that the most likely useful content is available locally. Any caching replacement policy could be employed for that matter (see the book by Michael Rabinovich and Oliver Spatschek for references13), and an evaluation of the best caching policy is outside the present scope.

We envision the caches being deployed and controlled by cloud storage providers directly. Moreover, we don’t assume any particular deployment topology. The location of caches in the network would be a choice of operators, and the only requirement is that clients must have routes to reach the caches. The extensions needed in Dropbox protocols to diverge traffic to caches, as well as other aspects related to the architecture’s implementation, are also out of our scope. Finally, we evaluate next whether the caching approach is cost-effective for storage providers, considering the cache costs and resulting bandwidth savings. Equally important questions, such as the privacy risks and the management overhead of deploying caches in several locations, are left for future work.

Evaluation Methodology

We evaluate the architecture using both our datasets and synthetic traces. We first use real traces to study whether storage caches are cost-effective in typical campus and ISP networks. Then, we rely on synthetic traces to extrapolate measurements and understand how costs and benefits vary when large populations are covered, or more sharing is seen in the network.

For the synthetic traces, we create an environment as in Figure 1. A number of devices act independently, performing updates that trigger
Measuring the Internet

Several downloads. A network cache is deployed to cover the devices. It intercepts updates, potentially serving content without involving the cloud. For simplicity, we assume all content is uploaded from the simulated environment — that is, external devices produce no workload to be downloaded.

We rely on CloudGen for creating synthetic traces. CloudGen is a synthetic workload generator for cloud storage services presented in our other work.\(^{10}\) CloudGen allows us to realistically reproduce the traffic created by arbitrary populations of Dropbox devices, based on parameters learned from traffic traces (for example, devices’ sessions duration). CloudGen generates a synthetic trace, indicating when each device is online, the updates in each namespace, and the data volume associated with each update.

We generate four types of synthetic traces:

- **Typical workload.** We set CloudGen to the number of devices observed in our datasets to illustrate how the synthetic workload compares to real traces.

- **High population.** We simulate large populations by tripling the number of devices — that is, roughly equivalent to having a Dropbox client in every IP address observed in Campus-1. Both upload and download volumes grow linearly with the number of devices.

- **High sharing.** We triple the mean number of devices sharing each namespace. The scenario mimics environments where users share lots of namespaces — for example, hypothetical companies adopting cloud storage as network file systems. Larger numbers of devices per namespace increase only download traffic.

- **High population and sharing.** We combine the previous two scenarios. Uploads grow linearly with the number of devices, while downloads grow as much as 20 times when compared to the typical scenario.

We calculate savings considering different cache sizes for both real and synthetic traces. We use the first month in each dataset of real traces to warm up the cache, and assess savings in subsequent months. For synthetic scenarios, we generate two-month-long intervals (warm up and evaluation) in 16 experiment rounds, and report mean values with 95 percent confidence intervals.

In all experiments, we use the simple and yet popular least-recently-used (LRU) policy: cache insertions and evictions are triggered by uploads; downloads manipulate the order of items (that is, updates) in the LRU cache, but don’t change cache contents. Finally, by tracking updates and cache behavior, we identify whether an update should be retrieved from the cache or cloud.

**Performance of the Cache-Based Architecture**

Now that we’ve detailed the architecture, let’s look at its performance.

**Bandwidth savings.** First, let’s consider the bandwidth savings obtained with our cache-based architecture, focusing on Campus-1’s real trace, as well as on several synthetic traces generated using CloudGen (parameterized according to that dataset). We omit results for the other datasets for the sake of brevity. Yet, conclusions in the following hold for all of the datasets.

We evaluate bandwidth savings using the byte hit ratio metric, which is computed as the volume of downloads served by the cache over the total volume of downloads reaching the clients. Figure 3a reports the byte hit ratios for two months extracted from the real trace (dashed purple curves) as well as a synthetic trace in the typical scenario (continuous black curve).

The differences between the two months in real traces are due to normal variations in the workload: month 3 presents higher volume in shared namespaces than month 2, which reduces the byte hit ratio, as we discuss next. Despite some divergences, the synthetic traces capture reasonably well the overall trend in real data. Such divergences are due to multiple factors. First, the synthetic trace is built by configuring CloudGen with input parameters extracted from the complete Campus-1 trace, as opposed to a subtrace corresponding to a particular month. Thus, it captures an overall trend observed during the whole three-month period, and not the particular behavior in shorter periods. Moreover, some simplifying assumptions made in the design of CloudGen contribute to the divergences, particularly for small caches. For instance, CloudGen assumes that devices behave independently, which might not be always the case. Multiple devices sharing namespaces might be online simultaneously, favoring the caching approach.

Yet, despite such divergences, we still see overall common trends. For example, for both...
real and synthetic workloads, even a small cache is able to remove a large fraction of avoidable downloads. In fact, a cache of 1 Gbyte would achieve from 21–56 percent of the byte hit ratio. Moreover, in all three curves, the byte hit ratio surpasses 90 percent with a 70-Gbyte cache.

Having compared real and synthetic traces in the typical scenario, we focus on the latter to evaluate the proposed architecture in other scenarios. Results are in Figure 3b. We note that, for any given cache size, the byte hit ratio is higher in the typical scenario than in any of the other three scenarios. This is because workloads are heavier in other scenarios, resulting in more cache misses. When the population size is increased, the larger upload volumes force the cache to remove old updates more often. In the high sharing scenario, the larger number of devices downloading content (that is, sharing namespaces) increases the probability that a device will fail to find a particular content in the cache, because the content has already faced eviction. Cache savings decrease even more in the high population and sharing scenario, as both factors are present. Nevertheless, despite such variations, our proposed architecture provides significant bandwidth savings in all scenarios. For example, for a 10-Gbyte cache, the byte hit ratio varies from 24–57 percent.

Overall, we conclude that even a modest cache can remove most of the Dropbox avoidable downloads when deployed in a typical network. As larger scenarios are considered, the cache size to achieve similar savings needs to be adjusted.

Cost-benefit tradeoff. The tradeoffs of deploying our architecture involve the cache costs and savings achieved by removing downloads from the cloud. In other words, an effective cache should prevent avoidable downloads at a minimum cost.
We evaluate costs and benefits of the architecture by computing the relative cost savings (rcs) for a time interval t:

$$rcs_t = \frac{cost_{\_nocache} - cost_{\_cache_{t_c}}}{cost_{\_nocache}}$$

where cost_{\_nocache} is the cost to serve all avoidable downloads in the time period t, while cost_{\_cache_{t_c}} corresponds to the cost associated with deploying a cache of size c bytes during time period t.

In short, rcs captures the fraction of the costs to serve avoidable downloads that the architecture can recover. The architecture is effective when rcs > 0; that is, the savings obtained by removing avoidable downloads at least pay off the costs associated with maintaining the cache. Note that rcs has an upper bound (rcs = 1) that indicates a scenario where all avoidable downloads are removed at negligible costs.

For the sake of simplicity, we express the cost_{\_nocache} and the cost_{\_cache} as functions of the number of bytes transmitted over the network and stored in the cache:

$$cost_{\_nocache} = d_t \times \alpha$$
$$cost_{\_cache_{t_c}} = m_{t_c} \times \alpha + c \times \beta,$$

where $d_t$ is the total number of bytes associated with avoidable downloads observed in the period t, $m_{t_c}$ is the number of bytes associated with avoidable downloads that a cache of size c bytes misses during time t, and $\alpha$ and $\beta$ represent bandwidth and storage prices per byte, respectively.

We take as reference for $\alpha$ and $\beta$ the prices for bandwidth and storage offered by Amazon Simple Storage Service and Elastic Compute Cloud (S3/E2C),\(^2\) which already include operational costs. Based on that, we evaluate rcs on two distinct cost setups, defined by the ratio $\beta/\alpha$: the price per byte of storage is a half of the bandwidth (that is, $\beta/\alpha = 1/2$), for example, because magnetic storage is used; and both prices are the same (that is, $\beta/\alpha = 1$), for example, SSD storage is taken. We use Amazon S3/E2C as reference because many storage providers rely on it to build their services. Other references (for example, market prices for SSD disks and mobile data plans) provide more optimistic cost-benefit tradeoffs.

Figures 3c and 3d show the rcs for the four scenarios and the two price ratios considering t equal to one month. The value of rcs increases to a maximum and then decreases. The inflection occurs when the best tradeoff between costs and benefits is achieved. Compared to Figure 3b, we see that the decrease in rcs happens despite higher byte hit ratios. Therefore, caches larger than a certain threshold add costs to the system, without providing significant savings.

Figure 3c ($\beta/\alpha = 1/2$) shows that rcs reaches 92–93 percent for typical and high population scenarios with cache sizes of 70 and 200 Gbytes, respectively. Scenarios with higher sharing achieve slightly better rcs. The maximum rcs reaches 95 percent for a 280-Gbyte cache in the high sharing scenario, going up to 97 percent in the high population and sharing scenario with a 500-Gbyte cache. Better tradeoffs with higher sharing are explained by the large volume of downloads: even if the byte hit ratio is lower than in typical scenarios for certain cache sizes (see Figure 3b for 100-Gbyte caches), the volume saved with avoidable downloads pays back the investment.

It’s worth noting the effects of the ratio between storage and bandwidth prices: the higher the storage price (compared to bandwidth), the lower the advantage of deploying the architecture. Comparing Figures 3c and 3d, we see that the maximum rcs slightly decreases for $\beta/\alpha = 1$; for example, it’s 7 percent lower than for $\beta/\alpha = 1/2$ in the typical scenario. Moreover, the costs of overestimated caches become equivalent to the bandwidth savings faster when storage costs are high.

In sum, our proposal cost-effectively achieves high bandwidth savings in a typical scenario, where 92 percent of the costs relative to avoidable downloads can be recovered. Storage providers thus have an incentive to deploy the caches, because savings by far compensate the costs. Benefits are higher with high-sharing (rcs reaches up to 95–97 percent), which hints to networks where cloud storage providers should start deploying the architecture.

We showed that content sharing in cloud storage services causes an expressive volume of avoidable traffic to providers. We proposed the use of network caches for content synchronization aiming at reducing this traffic. Our analyses suggest that even a small cache yields very high bandwidth savings. Such savings potentially cover the costs associated with building up the
The Impact of Content Sharing on Cloud Storage Bandwidth Consumption

Glauber Gonçalves is pursuing a PhD in computer science from Universidade Federal de Minas Gerais, Brazil. His research interests include computer networks, distributed systems, and performance modeling. Gonçalves has an MS in computer science from Universidade Federal de Minas Gerais, Brazil. Contact him at ggoncalves@dcc.ufmg.br.

Idilio Drago is a postdoctoral researcher at the Politecnico di Torino, Italy, in the Department of Electronics and Telecommunications. His research interests include Internet measurements, Big Data analysis, and network security. Drago has a PhD in computer science from the University of Twente. He was awarded an Applied Networking Research Prize in 2013 by the IETF/IRTF for his work on cloud storage traffic analysis. Contact him at idilio.drago@polito.it.

Ana Paula Couto da Silva is an associate professor at the Universidade Federal de Minas Gerais, Brazil. Her research interests include modeling and analysis of computer systems and energy efficient networking, characterization of Internet traffic, and complex network theory. Da Silva has a PhD in computer and system engineering from the Federal University of Rio de Janeiro. Contact her at ana.coutosilva@dcc.ufmg.br.

Alex Borges Vieira is an associate professor at the computer science department of Universidade Federal de Juiz de Fora, Brazil. His research interests include Internet measurements, Big Data analysis, and network security. Vieira has a PhD in computer science from Universidade Federal de Minas Gerais. Contact him at alex.borges@ufjf.edu.br.

Jussara M. Almeida is an associate professor of computer science at the Universidade Federal de Minas Gerais, Brazil. Her research interests include performance modeling of large-scale distributed systems. Almeida has a PhD in computer science from the University of Wisconsin–Madison. Contact her at jussara@dcc.ufmg.br.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

Acknowledgments

This research is partially funded by the FAPEMIG–PRONEX-MASWeb project – Models, Algorithms, and Systems for the Web, process number APQ-01400-14; by the National Institute of Science and Technology for the Web, CNPq, and FAPEMIG; and by the Vienna Science and Technology Fund through project ICT15–129, BigDAMA.

References


system, thus calling on cloud storage providers to consider the deployment of the architecture as a competitive advantage to their services. Future directions include analyzing privacy risks and the management overhead of deploying storage caches in different locations. This will require new models and simulations to support providers’ decisions on where to deploy caches.
IPv6 is an important component of the Internet's continued growth and evolution. It has grown exponentially and now carries nontrivial amounts of production traffic. Less well-understood is IPv6's topology and the way in which providers are using their IPv6 address allocations. Rather than relying on passive measurements or heuristics, the authors use uniform active probing; executing ICMP-Paris traceroute probes to an address in each /48 in all /32's advertised in the global IPv6 routing table (approximately 400 million traces). At this granularity, they characterize the distribution of IPv6 interface addresses in the wild, and find significant differences among providers and regions.
points. To make active probing feasible, CAIDA presently probes two addresses within each globally advertised IPv6 BGP prefix in each round of probing: the :1 address, and a random address. While this is an intuitive strategy to balance probing cost and time with expected coverage, to the best of our knowledge, neither its completeness nor its soundness have been rigorously examined.

Keeping this in mind, here we seek to inform two closely-related questions regarding Internet IPv6 topology: first, how are IPv6 providers using and subnetting their address allocations; and second, how can active measurement platforms more effectively and efficiently sample IPv6 topology? The basis of our analysis is a uniform ICMP-Paris traceroute probing of each /48 within all /32 prefixes advertised in the global BGP table (thus, exhaustive probing at a /48 granularity; 216 probes per /32). This first-of-its-kind dataset of approximately 400 million traces from 26 vantage points provides a valuable approximation of ground-truth to understand current IPv6 subnetting and allocation practice in the wild. While more granular (longer mask) IPv6 subnetting exists, /48’s were the recommended allocations to customers from a provider’s allocation for a decade before it became obsolete. Thus, while we haven’t exhaustively probed all possible subnets, we believe /48’s represent a reasonable compromise between completeness and probing volume/time. Our contributions thus include the following:

- an analysis of active Paris-traceroute probing of an address in each /48 within all globally advertised /32 IPv6 prefixes (this dataset, gathered in collaboration with CAIDA, is publicly available);
- a characterization of Internet-wide IPv6 allocations, subnetting, and adherence to recommended best common operational practices;
- an analysis of per-provider and per-regional differences in IPv6 subnetting; and
- the distribution of discovered IPv6 interfaces.

Our hope is that this work serves to inform both the development of future efficiency-optimized active IPv6 probing algorithms, as well as the community’s understanding of provider use of IPv6 address allocations.

**Background**

The growth, use, and adoption of IPv6 has been extensively measured and studied. Recently, Jakub Czyz and his colleagues found significant differences in the adoption of IPv6 in a longitudinal study across 10 different datasets, each representing a different facet of IPv6 (for example, use of IPv6 in the DNS, routing, and traffic). Rather than presenting a broad study of adoption, we seek to more deeply understand a single aspect — IPv6 subnetting — in a single, Internet-wide snapshot.

Prior work examines IPv6 topology, but is largely limited to the AS-level topology as observed in BGP routing announcements. For example, Amogh Dhamdhere and his colleagues examine the congruence of IPv4 and IPv6 AS paths, IPv6 AS path lengths, and the most central IPv6 ASes. In contrast, our work studies properties of the interface-level IPv6 address allocation, and statistical properties revealed via active probing.

Related to our work is the spatial classification of IPv6 addresses observed by a large content distribution network (CDN). By clustering the addresses of Web clients that access the CDN, inference can be made on the ways in which providers are allocating addresses and subnets to clients. Our work is largely complementary to this prior study: rather than opportunistically relying on passive traffic (such as clients that access the CDN) our active probing helps eliminate possible sample bias. However, our technique limits us to understanding the addressing at a /48 granularity, while passive techniques can reveal more fine-grained details.

To the best of our knowledge, CAIDA performs the only continuously maintained active IPv6 topology discovery platform built on its Ark platform. Each vantage point in Ark takes routed IPv6 prefixes (as viewed from the global BGP table) as input, and probes the following: the :1 station address within each prefix; and a random station address within each prefix. Probes consist of ICMP-Paris traceroutes performed by the scamper packet prober.

We're motivated in part by previous work that performed active topology probing of the IPv6 Internet by proposing heuristics and techniques to cope with the address space's size. A central problem, however, is obtaining a basis for evaluating the performance of such intelligent IPv6 active probing — without ground-truth of the possible topology that could be discovered, only relative metrics are possible when evaluating topology probing systems. As such, we hope our work serves to inform the development of future efficient IPv6 active network mapping algorithms.
In addition, our work sheds some light onto the way in which providers are using IPv6 addresses for infrastructure, and insight into the subnets they might be allocating. Such operational insight is interesting with respect to published best common operating practice guidelines, which recommend the following, for instance: dedicating the first or last /48 per region to number infrastructure, numbering point-to-point interfaces out of /64 prefixes, not subnetting on non-nibble boundaries, and creating subnet prefixes of equal size. We find distinct evidence in practice of both adherences to, and deviations from, these recommendations.

**Methodology**

Because we take a provider-centric view, our work centers on studying prefixes at the /32 granularity (that is, IPv6 prefixes with 32 bits of network mask). We start with the set of globally advertised IPv6 BGP prefixes, as visible from routeviews (www.routeviews.org), and limit our examination to advertised /32 prefixes. Future work should subdivide larger prefixes (those with masks less than 32) into constituent /32s.

To perform the active topology probing, we rely on scamper, an advanced packet prober that implements a variety of traceroute methods. We use scamper to send Time-to-Live (TTL) limited ICMP6 probes, where the probe headers are formed using the Paris traceroute technique such that each probe takes the same path, even over flow-balanced paths. The probes elicit ICMP6 TTL-exceeded messages, where the source address corresponds to the router's interface used to reach the prober. Thus, we recover the set of router interface IPv6 addresses on the forward path toward the destination.

To interrogate every /48 in the set of globally advertised /32 IPv6 prefixes visible from routeviews, we use 26 IPv6-capable vantage points from the CAIDA Ark infrastructure. The Ark vantage points are physically globally distributed, as well as connected to a diverse set of IPv6 networks. We distribute the task to issue IPv6 ICMP-Paris traceroutes to the ::1 address in every /48 (one traceroute to each destination) across these vantage points. In total, our data includes one-time probing toward approximately 408 Million /48s. We chose the ::1 address due to its general popularity as the subnet gateway router interface address. While the tools and platform are the same, this methodology is quite different from CAIDA’s routine IPv6 probing, discussed in the “Background” section, which uses scamper-equipped Ark nodes to probe one random address in each of the roughly 16,000 advertised prefixes (regardless of prefix length) every three days.

Scamper terminates probing toward a destination after reaching a gap limit of five successive unresponsive hops, or if a loop is detected (where the same IPv6 address is observed in response to two different probe TTLs). These settings mirror CAIDA’s default configuration for their continuous probing, thereby helping to mitigate comparison bias between the two datasets.

As we mentioned, the entire dataset collected and analyzed in this article is publicly available from CAIDA. So not only should our results be repeatable, we hope this dataset enables future research and analysis of IPv6 topology.

**Probing Details**

Due to the volume of probing required, probing was performed from 13 November 2014 to 1 March 2015. The set of prefixes to probe is based on the information available at the beginning of this time period, at which time there were 6,162 /32s. The assignment of probes to vantage-points is pseudo-random. The list of 32-bit prefixes is shuffled randomly and then split into batches of 110 prefixes. The 110 × 216 /48s in each batch then is shuffled randomly and split into fixed-size chunks. Each chunk is assigned round-robin to a particular vantage point for probing.

We distributed the probing work among vantage points to spread the probing load and decrease the total time required. While different vantage points will take different paths to reach a particular destination, each vantage point probes a random and large subset of the total destination set. For a given traceroute, newly discovered interfaces primarily come from hops past the vantage point’s local neighborhood, which is already well-discovered over the course of probing. Thus, the net impact of using multiple vantage points on the discovered interface graph is likely to be negligible.

Out of the 407,793,780 probes issued, only 137,235 (0.03 percent) reached their destination—an unsurprising result due to the currently sparse IPv6 Internet population. We also determined that 35,118,396 probes (8.6 percent) were terminated due to loop detection, and another
Empirical Study of Router IPv6 Interface Address Distributions

203,837,347 (50 percent) due to gap limiting (too many unresponsive hops). The remainder of the probes terminated with some other ICMP6 unreachable code. All of this probing results in a set of 240,155 unique IPv6 addresses, belonging to 152,481 unique /48 subnets, in 4,763 unique /32s prefixes, belonging to 4,173 different ASes. These 240,000 IPv6 addresses consist of 137,235 unique responsive destinations, and 128,804 unique intermediate router IPv6 interface addresses (where a responsive destination in one trace might also appear as an intermediate hop in a second trace).

Address Population
This population study is primarily concerned with the allocation and use of IPv6 addresses for numbering router interfaces. IPv6 router address discovery is important to building topology maps and understanding critical infrastructure. While it’s difficult to discover responsive IPv6 hosts (our traceroutes empirically yield only a 0.03 percent response rate), by sending probes to destinations uniformly across each /32, we hope to elicit responses from as many routers along the forwarding paths as possible. For this reason, we also consider the intermediate traceroute hops in our results. This reveals additional interfaces both within the /32 being probed, as well as in other /32s in use by transit providers. Naturally, we discover some of the same interfaces appearing in many traceroutes, but duplicates are removed and only unique interface addresses retained in the population. However, the effect of using all responding interfaces in our results is that the effective sampling rate isn’t uniform across all /32s; those in use by transit providers receive a much higher sampling rate. To account for this, in the following section we consider not only aggregated statistics, but individual /32 distributions, and their contribution to the overall statistical results.

Table 1. Topology discovery statistics – uniform probing versus existing Center for Applied Internet Data Analysis (CAIDA) methodology.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Uniform</th>
<th>CAIDA (All)</th>
<th>CAIDA (/32’s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traces</td>
<td>407,793,780</td>
<td>1,059,058</td>
<td>302,531</td>
</tr>
<tr>
<td>– Destination reached</td>
<td>137,235 (0.03%)</td>
<td>119,052 (11.5%)</td>
<td>33,728 (11.1%)</td>
</tr>
<tr>
<td>– Looped</td>
<td>35,118,396 (8.6%)</td>
<td>81,779 (7.7%)</td>
<td>19,331 (6.4%)</td>
</tr>
<tr>
<td>– Gapped</td>
<td>203,837,347 (49.7%)</td>
<td>527,822 (49.8%)</td>
<td>145,687 (48.2%)</td>
</tr>
<tr>
<td>Interfaces</td>
<td>128,804</td>
<td>57,455</td>
<td>29,882</td>
</tr>
<tr>
<td>Edges</td>
<td>267,647</td>
<td>144,311</td>
<td>68,801</td>
</tr>
</tbody>
</table>

Results
Here, we present a relative comparison of our probing against CAIDA’s current IPv6-wide active topology probing, analyze the distribution of interfaces within /32s, and characterize difference among /32s observed in the wild.

Relative Comparison
We first seek to understand the relative difference between our uniform probing, and the current state-of-the-art collection from CAIDA. Our objective is to obtain a sense of how much topology information is gathered today, versus how much could be obtained through more exhaustive and/or intelligent probing.

Table 1 compares our uniform probing against one cycle of CAIDA’s production IPv6 probing collected on 1–2 March 2015 from 26 vantage points (both our method and CAIDA’s production method use the same vantage points). In addition, we analyze a subset of the CAIDA traces that corresponds to exactly those /32’s we probe. Because CAIDA probes destinations within all routed prefixes, this restricted dataset is a more meaningful comparison by using the same set of /32 prefixes.

We observe approximately the same fraction of gap limited and looped traces across all three datasets, although more traces loop in our uniform probing, presumably because we target destinations for which there’s no more specific route. The CAIDA traces reach a much higher fraction of destinations due to the fact that there’s frequently a “::1” address associated with each routed prefix, as opposed to our probing of largely unused space.

We then examine the number of unique interfaces (router interface IP addresses) and edges (IP address pairs in successive traceroute hops) obtained in each dataset. CAIDA’s probing of the /32s finds only 23 percent of the number of interfaces our uniform probing discovers,
Measuring the Internet

Figure 1. All of the IPv6 interfaces discovered, sorted by the /32 prefix. Vertical jumps in the plot show concentrations of interfaces discovered in particular /32 prefixes. Réseaux IP Européens Network Coordination Center (RIPE NCC) is dominant, both in the number of active prefixes and in the fraction of interfaces discovered. (AFRINIC = African Network Information Center, APNIC = Asia-Pacific Network Information Center, ARIN = American Registry for Internet Numbers; IANA = Internet Assigned Numbers Authority, and LACNIC = Latin America and Caribbean Network Information Center.)

and only 26 percent of the number of edges from uniform probing. However, this coverage comes at the cost of greater than 1,000 times more probes.

These high-level findings suggest that there’s a significant amount of currently undiscovered (on a per-probing round basis) topology, yet the cost of discovering this topology with current methodology is prohibitively high.

Distribution of Interfaces within /32s

We next examine the distribution of sources (router interfaces) replying to our probing (inclusive of intermediate router hops), as organized by the /32 to which the response’s IPv6 source belongs. Looking at the number of unique unicast IPv6 router addresses within each /32, we see a wide variance (see Figure 1). Here, the y-axis is the cumulative fraction of total discovered unique IPv6 interfaces, while the x-axis is simply a sequential identifier for each /32 (assigned in increasing order of network prefix). Note that the total number of observed /32s is fewer than 6,162 (the number of /32s we probe), because our probes didn’t elicit responses from every /32 targeted. Further, we receive responses from /32s not in the original set of 6,162. Vertical jumps in the plot show concentrations of interfaces discovered in particular /32 prefixes. We also show via colored banding the regional Internet registry (RIR) to which each prefix belongs. This lets us see the dominance of Réseaux IP Européens (RIPE), both in the number of active prefixes and in the fraction of interfaces discovered. Table 2 lists the specific /32 prefixes in which the most interfaces were discovered. Comparing Table 2 to Figure 1, we can easily see the contributions of the densely populated /32s from Bitcanal and XS4ALL, as well as some of the other large interface blocks from major providers. Surprisingly, the majority of the ASes represented in Table 2 aren’t tier-1 transit providers, and some have relatively low IPv6 AS rank.21 (We based this information on the CAIDA IPv6 Org Rank dataset from 1 September 2014, which is the most recent available at the time of this writing.)

In Figure 2 we show the same data, this time with the x-axis aligned by the /32 prefix itself (as opposed to the sequential index of the /32). This view lets us observe the relative population of the few /8s from which allocations are currently distributed, with 2000::/8 and 2a00::/8 making up the lion’s share, and 2600::, 2400::/8, and 2800::/8 a distant third, fourth, and fifth, respectively. In terms of regional allocations, RIPE, American Registry for Internet Numbers (ARIN), Asia-Pacific Network Information Center (APNIC), Latin America and Caribbean Network Information Center (LACNIC), and African Network Information Center (AFRINIC) all contribute to the 2000::/8 block, while the sole 2400::/8 allocation belongs to APNIC, all three 2600:: allocations belong to ARIN, and the only allocation from 2800::/8 is to LACNIC. The large number of interfaces in the 2a00::/8 block are in RIPE allocations, with a few AFRINIC interfaces next door in the 2c00::/8 allocation.

For the last component of our look at the overall population of /32s, we again rearrange the x-axis, this time sorting the prefixes from high to low by their interface count (see Figure 3). This plainly shows 90 percent of the observed interfaces coming from 10 percent of the prefixes.

Example /32 Interface Distributions

Having surveyed all of the /32s corresponding to each discovered interface, we then drill down to examine the distribution of responding interfaces within individual /32s. Figure 4 shows 10 of the top 12 /32 prefixes by census, and it’s clear that IPv6 address allocation schemes vary
Within AS 43447 (ORANGE-PL) we find nearly 8,000 interfaces uniformly distributed between 2a00:f42:0:: and 2a00:f42:2000::, or one per /48, while the other 47,000 /48s are unpopulated. Hurricane Electric on the other hand shows thousands of interfaces within the first and second /48 (2001:470:0:: and 2001:470:1::), and another grouping of thousands of interfaces between 2001:470:1f04:: and 2001:470:1f14::, Hurricane also has many interfaces distributed across the remainder of this /32. Looking further down the graph, we see a number of /32s (belonging to Tinet, Cogent, DE-CIX, and Level 3), where more than 1,000 interfaces appear in the first /48 and far fewer or no interfaces are discovered in the rest of the /32. We observe that this distribution seems to reflect current best practices for IPv6 address allocation, which prescribes allocating subnets from the beginning of the allocation.17 Other /32s (belonging to Easynet, TANet, and Snap Internet) have blocks of responding interfaces more widely distributed across their /32. Last, we see a /32 belonging to NetCologne, with more than 2,000 interfaces responding in the 2001:4dd0:ff00:: /48 at the upper end of the /32, and with negligible allocations elsewhere in the range. This set of /32s illustrates

Table 2. Top /32 prefixed by number of unique interface addresses discovered.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interfaces</th>
<th>Autonomous system (AS) no.</th>
<th>Organization</th>
<th>Organizational rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a00:4c87::</td>
<td>62,392</td>
<td>197426</td>
<td>Bitcanal</td>
<td>7,281</td>
</tr>
<tr>
<td>2001:980::</td>
<td>40,860</td>
<td>3265</td>
<td>XS4ALL</td>
<td>2,217</td>
</tr>
<tr>
<td>2a00:f42::</td>
<td>7,869</td>
<td>43447</td>
<td>Orange</td>
<td>116</td>
</tr>
<tr>
<td>2001:470::</td>
<td>6,886</td>
<td>6939</td>
<td>Hurricane Electric</td>
<td>9</td>
</tr>
<tr>
<td>2406:e000::</td>
<td>5,873</td>
<td>23655</td>
<td>Snap Internet</td>
<td>525</td>
</tr>
<tr>
<td>2001:288::</td>
<td>4,503</td>
<td>1659</td>
<td>TANet</td>
<td>1,739</td>
</tr>
<tr>
<td>2001:4dd0::</td>
<td>2,708</td>
<td>8422</td>
<td>NetCologne</td>
<td>552</td>
</tr>
<tr>
<td>2001:6f8::</td>
<td>2,656</td>
<td>4589</td>
<td>Easynet</td>
<td>397</td>
</tr>
<tr>
<td>2001:668::</td>
<td>2,356</td>
<td>3257</td>
<td>Tinet</td>
<td>4</td>
</tr>
<tr>
<td>2001:550::</td>
<td>2,183</td>
<td>174</td>
<td>Cogent</td>
<td>2</td>
</tr>
<tr>
<td>2001:1900::</td>
<td>2,172</td>
<td>3356</td>
<td>Level 3</td>
<td>1</td>
</tr>
<tr>
<td>2001:7f8::</td>
<td>2,065</td>
<td>6695</td>
<td>DE-CIX</td>
<td>7,281</td>
</tr>
</tbody>
</table>

Figure 2. All of the IPv6 interfaces discovered, indexed by the /32 prefix. This view lets us observe the relative population of the few /8s from which allocations are currently distributed.

Figure 3. All of the IPv6 interfaces, with a reverse-sorted number of interfaces per /32 (sorting from high to low). Ninety percent of the observed interfaces come from 10 percent of the prefixes.
the wide variety of IPv6 address allocation strategies observed in the wild.

We don’t plot the top two /32 prefixes (listed in Table 2), because nearly every /48 is represented, resulting in interface counts that are an order of magnitude higher than the rest of the top 10 and appear simply as linear diagonal lines when plotted. It’s not clear why these two ASes have so many interfaces responding.

While this sampling of the top responding /32s highlights the wide variance in addressing strategies, what we’re really interested in discovering is if there are commonalities between different organizations’ address-allocation schemes. For this we look to several plots showing local prefix distributions. Not to be confused with the IPv6 prefix reserved for link-local addressing, the local prefixes we refer to here are the 33rd through 48th bits of the network address. These are local subnet prefixes given an assumed /32 prefix allocation.

We start with a histogram of /48 local prefix bit patterns shown in Figure 5, along with Table 3, which shows the exact values for the top 10 peaks from the histogram. We see that more than half of the /32s probed respond from the X::0::/48, and nearly one-fourth respond from the X::1::/48, which is consistent with the sample /32s we observe in Figure 4. Perhaps more interesting is the distinct spike we observe at X::8000::/48 and the somewhat lower spike at each /36 increment (X::1000::/48, X::2000::/48, X::3000::/48, and so on). We
think this is likely indicative of hierarchical sub-netting (/34 and /36) taking place within the /32s, but could also reflect human preference for memorable subnet prefixes. We defer distinguishing the root cause of these patterns to future work.

We further break down the distribution of interfaces by RIR assignment in Figure 6, which shows significant differences between the regions. Table 5 gives a detailed breakdown of all interfaces discovered by RIR allocation. In comparison with Figure 4, we observe some similar patterns, in that the majority of the RIRs carry a large fraction of their interfaces within the first /48. RIPE is the most notable exception to this, but we find that the RIPE distribution is dominated by two /32s (2a00:4c87::/32 and 2001:980::/32) that contribute 103,252 of RIPE’s 171,308 interfaces. In general, this reinforces the notion that the relative sparsity of the IPv6 Internet infrastructure, combined with the scale of individual allocations, allow a few actors to dominate the statistics of the whole. Additional comparisons between Table 5 and Table 2 show that it’s likely a few major ASes dominating the /48 distribution in each of the regions.

Note that for Figures 5 and 6, we count only unique /48 bit patterns, not individual interfaces. The reason for this is that the interface count in the X:X:0::/48 is so large (Table 4) that it made distinguishing patterns in the lower volume /48s difficult, even with a log-scale plot. Given that our probing strategy was only comprehensive to the granularity of /48 prefixes, it makes sense to aggregate to that level for statistical analysis. That being said, we also want to understand why there’s such a disproportionately large number of interfaces recorded with the X:X:0::/48 bit pattern (keep in mind that we only probed one address in each /48; discovering 10 interfaces for every responding X:X:0::/48 was unexpected), so we perform further analysis of the paris-traceroutes where they were discovered. We find that on average every trace with at least one such interface contains five interfaces matching this pattern, and they appear in the middle of the traces. Looking at the /32 prefixes from the interfaces in question, we find that the vast majority belong to major transit ASes (Tinet, Cogent, Orange, Hurricane Electric, and Nippon Telegraph and Telephone [NTT]) and some large ISPs (Time Warner and Comcast). Comparing these observations with Figure 4 leads us to the conclusion that large ISPs are more likely to assign IPv6 addresses densely from the start of their /32 allocation. A more comprehensive examination of the allocation of individual router interface host addresses is available elsewhere.22

Observations and Analysis
At its heart, our study is a population survey; however, it yields interesting insights and presents numerous avenues for follow-on study.

One unexpected result of this study is that the top 12 most populous /32s we discovered aren’t primarily composed of the largest IPv6 customer-cone AS rank networks. Several are represented in our list, but they’re offset by an equal number of low-ranked organizations. Looking at these most populous /32s reveals widely differing allocation schemes. Some have interfaces spread quasi-uniformly throughout the range (apparently by preference, not necessity), while others have them clustered exclusively at the beginning or end of the range. However, when we look only at the distribution of /48s, aggregated across all /32s, some highly distinct patterns become visible. Interfaces are clustered at common subnet boundaries — for example, /33 (X:X:8000::), /34 (X:X:4000::; X:X:c000::), /35 (X:X:2000::; X:X:6000::; and so on), and /36 (X:X:1000::; X:X:3000::; X:X:5000::; and so on). At the same time, it’s also clear that human preference plays a large role, with 1, 2, 3, 100, ff, 4, and 10 outranking any of the aforementioned prefixes in popularity (also observed in Matthew Gray’s work22). We also note that our traceroute-based
### Measuring the Internet

<table>
<thead>
<tr>
<th>Table 5. Interface distribution by RIR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIR</td>
</tr>
<tr>
<td>RIPE</td>
</tr>
<tr>
<td>ARIN</td>
</tr>
<tr>
<td>APNIC</td>
</tr>
<tr>
<td>LACNIC</td>
</tr>
<tr>
<td>AFRINIC</td>
</tr>
<tr>
<td>6to4</td>
</tr>
<tr>
<td>IANA</td>
</tr>
</tbody>
</table>

methodology primarily collects addresses assigned to outward-facing router interfaces, which might skew the results we observe.

Although our methodology results in a reasonable sample size, when compared with the potential population of a single /32, it’s still quite small. This allows overall statistics to be dramatically skewed by one or two ASes and requires us to examine the results at a finer granularity. Specifically, Bitcanal and XS4ALL both have responding interface counts an order of magnitude higher than any of the tier-1 service providers, and account for over 40 percent of our total results. This dominates any aggregate statistic they’re included in, particularly those for the RIPE RIR, to whose allocation they both belong. This is definitely an area we would like to investigate further, both to find out why those particular ASes respond in this way, as well as to determine aggregate analysis metrics that are meaningful in the context of the massive IPv6 population disparity that currently exists, and is likely to continue to exist, for many years to come.

By its nature and design, this work has raised as many new questions and research directions as it has answered. For instance, our exhaustive probing provides only a snapshot in time. It would be valuable to perform a longitudinal study to understand how the infrastructure IPv6 address population distributions shift over time. Further, while validation is difficult, we plan to solicit feedback on our findings from willing providers where possible.

Due to the volume of probing required, we also chose to limit this study to the granularity of /48s, but it’s well-known that major providers are allocating subnets at a finer granularity than this, so more granular probing might also be beneficial, whether wide-scale or on selected prefixes. At the other end of the scale, the present work doesn’t consider prefixes larger than /32 (for example, with masks greater than 32). While there are many more prefixes with 32 bits of mask than those with masks less than 32 bits, these larger prefixes represent large providers that we haven’t yet fully characterized. We do this so as to scope our probing effort, while facilitating comparisons among provider /32s. In the future, we plan to subdivide these large aggregates into their constituent /32s, and include them in our uniform probing.

While our ultimate goal is understanding the IPv6 topology, this work is largely restricted to characterizing the population of IPv6 addresses discovered via exhaustive active probing. Future work can examine the same dataset\textsuperscript{12} to make stronger topological inferences. For instance, if two probes to contiguous /48s are topologically congruent, this could imply that the /48s are part of a larger routed aggregate. In contrast, if paths to these two /48s reveal different router interfaces, we could more strongly infer that there exist distinct routed subnets for these two destinations.

In previous work, we showed the benefits of intelligent probing algorithms to improve efficiency (number of interfaces discovered/number of traceroutes required) in the IPv4 Internet,\textsuperscript{23} but have struggled to apply these same algorithms to IPv6. We hope that the results of this exhaustive probing study can lead to more efficient IPv6-optimized probing algorithms. Last, we again note the large impact on our population statistics caused by only two /32s (belonging to Bitcanal in Portugal and XS4ALL in the Netherlands) that appeared to have nearly every /48 populated. Neither of these organizations is a highly ranked ISP (by the IPv6 customer-cone metric), so it strikes us as odd that they would have an order-of-magnitude larger interface population responding than organizations like Level 3 and Hurricane Electric. We would like to explore this finding further, to identify the allocation techniques in use at those organizations.

**Acknowledgments**

We thank the anonymous reviewers for their valuable comments, Young Hyun for his assistance in executing the exhaustive probing, Kimberly C. Claffy (‘kc claffy’) for her support, and CAIDA for the Ark infrastructure. This work was supported in part by US National Science Foundation grant CNS-1111445 and US Department of Homeland Security Cyber Security Division contract N66001-2250-58231. Views and conclusions are those of the authors and...
shouldn’t be interpreted as representing the official policies, either expressed or implied, of the US government.

References


Justin P. Rohrer is a research assistant professor in the Department of Computer Science at the US Naval Postgraduate School (NPS), where he leads the Tactical Networked Communication Architecture Design (TanCAD) lab. His research focuses on tactical networking, including resilient and survivable mobile transport and routing protocols, and measuring network resilience. Rohrer has a PhD (with honors) in electrical engineering from the University of Kansas. Contact him at jprohrer@nps.edu.

Blake LaFever is a lieutenant commander in the United States Navy. His research interests include military cyber defense, information assurance, and network operations. LaFever has an MS in cyber systems and operations from the US Naval Postgraduate School. Contact him at bwlafeve@cmand.org.

Robert Beverly is an assistant professor in the Department of Computer Science at the US Naval Postgraduate School. His research in computer networks focuses on network architecture, large-scale traffic analysis and data mining, measurement, and security. Beverly has a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology. Contact him at rbeverly@nps.edu.
By leveraging the uneven distribution of traffic among network flows, here the authors improve the query throughput of Cuckoo hashing. They achieve this by placing the most frequently used items in the table that’s accessed first during the Cuckoo query operation. Their scheme is named Cuckoo cache, as it’s conceptually similar to a cache but implemented inside the Cuckoo hash with little additional cost. Cuckoo cache is evaluated using a traffic monitoring application fed with real traces. The results are compared with existing Cuckoo hashing implementations and show significant improvements.

One of the functions widely used in networking is flow identification that’s employed for different purposes such as routing, security, quality of service, and traffic monitoring. Several methods have been proposed to efficiently perform flow identification at high speed. When flow identification is performed on an application-specific integrated circuit (ASIC) chip, one option is the use of a content-addressable memory (CAM). Due to CAMs’ poor scaling performance and high power consumption, several hash-based structures have been proposed as an alternative. Because these structures use standard memories, they also can be implemented easily in both software and hardware field-programmable gate array (FPGA)-based network equipment.

Hash-based structures are able to provide better scalability than CAMs, but suffer from several drawbacks, mainly because of the need to resolve collisions that can occur when the key to retrieve is hashed to identify the memory location in which the value is stored. Drawbacks can include a poor memory-occupation ratio, worst-case query time, and failure probability when a new element is inserted on the table. In the literature, many hash-based structures have been proposed to overcome these limitations. One of the most successful ones is the Cuckoo hashing scheme.

Cuckoo hashing is a multiple hash table scheme in which the inserted elements can be moved among the tables to make space for new items. This scheme provides a fixed time to perform
the query operation and high memory usage at the cost of making the insertion procedure more complex than for other schemes. The features of Cuckoo hashing make it attractive for high-performance packet-processing platforms.8,9 To be efficient, the number \( d \) of hash tables needed to implement the Cuckoo hashing is at least \( d = 4 \) when the number of elements per hash table position is \( b = 1 \). In fact, with four tables, Cuckoo hashing can operate effectively when memory use exceeds 95 percent. Occupancies close to 100 percent also can be achieved with fewer tables if \( b \) is larger than 1.10 However, in general, the increase required in \( b \) to achieve a similar occupancy is larger than the reduction in \( d \). For example, with \( d = 2 \), at least \( b = 4 \) is needed to achieve an occupancy close to 95 percent. This increases the memory bandwidth, as accessing the first table requires reading four elements, which is the worst case for \( d = 4 \) and \( b = 1 \). Additionally, reading a table in one memory access also requires a 4x wider memory interface (that corresponds to an increased pin count and to a bigger wire placement complexity). This is a limiting factor on FPGA implementations using external memory. As an example, the last generation NetFPGA11 board hosts an external SRAM memory connected to the FPGA with a 3 x 36-bit bus. The case \( d = 4 \) corresponds to a worst-case query time of four memory accesses and, if we assume that the keys are equally distributed, we can estimate an average query time of 2.5 memory accesses.12

Reducing the average query time has two positive effects. First, it improves the throughput of Cuckoo hashing, allowing processing more packets per second and therefore enabling the monitoring of higher speed links. Second, it reduces the energy consumption related to each query. In fact, as we discuss in other work,12 the dynamic power consumption related to the memory access represents a significant part of the energy spent by Cuckoo hashing. This is highly pertinent, because energy efficiency has become an important design goal for networking equipment.13 Our idea to reduce the average query time of Cuckoo hashing exploits the uneven distribution of Internet traffic.14,15 In fact, a large portion of the packets traveling the network belong to relatively few flows. Therefore, storing these flows in the first table that’s accessed by the query operation allows reducing the average query time. Here, we present an algorithm that moves the frequently queried items into the first table, moving away from the first table the items that aren’t frequently used.

**Cuckoo Hashing**

The data structure used in Cuckoo hashing is a set of \( d \) hash tables such that an element \( x \) can be placed in tables 1, 2, ... \( d \) in positions \( h_1(x) \), \( h_2(x) \), ..., \( h_d(x) \) given by a set of \( d \) hash functions. In general, each position can store up to \( b \) elements, but in the remainder of this article we assume that \( b = 1 \). The following operations can be performed on the structure:

- **Query(\( x \)).** The table given by \( i = 1 \) is selected and position \( h_i(x) \) is accessed and compared with \( x \). If there’s no match, the second table is selected and the process is repeated. If no match is found in that table, the third table is selected and the search continues until a match is found or until all \( d \) tables have been searched. If the item \( x \) is found in one of the tables, the query operation is stopped and the value \( v_x \) associated to \( x \) is provided, otherwise a Not Found value is given as output.

- **Update(\( x, v_x \)).** First, the query operation is performed. If the item \( x \) is found, the value \( v_x \) associated to \( x \) is updated, otherwise the insert operation is performed.

- **Insert(\( x, v_x \)).** The table given by \( i = 1 \) is selected and position \( h_i(x) \) is accessed. If it’s empty, the new element is inserted there. If not, the second table is selected and the process is repeated. If the position in that table isn’t empty, the third table is selected and the search continues until a table with an empty position is found or until all \( d \) tables have been searched. At that point, a random table \( j \) is selected and the new element \( x \) is stored in position \( h_j(x) \). Then, the insertion process is executed for the entry \( y \) that was displaced when inserting \( x \), but not considering table \( j \) as an option for insertion. This procedure is recursive and tries to move elements to accommodate the new element if needed.

- **Remove(\( x \)).** This is the same operation as a query, but when the element is found it’s removed.

Several architectural options can be used to implement Cuckoo hashing. The most direct implementation is to store the \( d \) hash tables in
a single memory. Another option is to have \( d \) memories such that each table is stored in a different memory and all the memories are accessed in parallel. For the case of a single memory, accesses can be sequential or concurrent, depending on the implementation. When the system is based on a modern CPU, concurrent access is common and coordination among the threads that use the Cuckoo hash is an important issue. On an FPGA implementation, it’s common to store the tables on external memory. This is because it can take several megabytes to store the required memory (for example, in one paper the authors estimate 8 Mbytes for a flow counter table and 6.5 Mbytes for the flow key table), and thus exceed the capacity of the internal memories inside the FPGA. In many FPGA systems, packet processing takes place in a pipeline such that several packets are processed at the same time in the different stages. In many such systems, the biggest bottleneck occurs when the memory is accessed to retrieve the data needed to process the packets. Our enhanced Cuckoo cache reduces the number of memory accesses — and consequently the severity of this bottleneck — thus increasing the maximum achievable throughput.

Another difference between both cases is that for a CPU implementation, the memory hierarchy of the CPU with the different cache levels has an important effect; while for an FPGA-based implementation, no cache or memory hierarchy is used. In the remainder of this article, we present the proposed algorithm focusing on a single memory architecture with sequential access, as is the case in most FPGA-based network equipment. The use of the algorithm in a modern CPU is also briefly discussed and evaluated in terms of execution time and cache use.

**Cuckoo Cache**

The proposed improvement of Cuckoo hash is achieved by moving the most frequently accessed items to the table that’s accessed first during queries. The following operations are defined and describe how the proposed structure works:

- **Query**(x). This is similar to the standard query operation, but if the item is found, a counter value \( c_x \) is incremented.
- **Update**(x, \( v_x \)). First, the query operation is performed. If the item \( x \) is found, the counter value \( c_x \) is incremented and value \( v_x \) associated to \( x \) is updated. If the item isn’t found, the insert operation is performed.

- **Insert**(x, \( v_x \)). This is similar to the standard insert operation, but when the item is inserted, the counter value \( c_x \) is set to 0.
- **Conditional insert**(x, \( v_x \)). First, the operation searches in all \( d \) tables. As soon as a table is empty, the element is placed there. If all \( d \) tables aren’t empty, a random table \( j \) is selected. If \( j = 1 \), before removing the item \( y \) stored in the memory location \( h_1(x) = h_1(y) \), a comparison between \( c_x \) and \( c_y \) is performed. If \( c_x > c_y \), \( x \) is inserted in the first table and the conditional insert is executed for the entry \( y \). Instead, if \( c_y < c_x \), \( x \) is inserted in the second table and the conditional insert is executed for the entry \( x \) stored in the location \( h_2(x) = h_2(y) \). When \( j = 1 \), the element \( y \) is removed, \( x \) is placed there, and a conditional insert is executed for the entry \( y \).

- **Conditional query**(x). First, the query operation is performed. After, the item \( y \) stored in the memory location \( h_1(x) = h_1(y) \) is checked. If \( c_x > th + c_y \), \( x \) is moved in the first table and a conditional insert of the item \( y \) is performed. Here \( th \) is the threshold value that triggers the movement of the most frequently used items.

- **Remove**(x). This is similar to the standard remove operation.
- **Scrub()**. For each element \( x \) stored in the \( d \) tables, compute \( h_1(x) \), and read the item \( y \) stored in the memory location \( h_1(x) = h_1(y) \). If \( c_x > th + c_y \), move \( x \) in the first table and perform a conditional insert of the item \( y \).

The conditional query operation is designed to move the most frequently accessed items in the first table, while the conditional insert allows moving items among the tables while maintaining in the first table the most frequently queried items.

The conditional query operation can require more memory accesses than the standard query operation. However, because after the conditional query the most frequently used items are shifted to the first table, the overall average number of memory accesses will decrease. The \( th \) value is used to trigger the movement of the most-used items. Higher values of the threshold reduce the number of reallocations, thus decreasing the probability that a conditional query triggers a sequence of memory...
Cuckoo Cache: A Technique to Improve Flow Monitoring Throughput

movements. On the other hand, if the threshold value is too high, the benefit of reallocation is reduced to only a few items that have the highest frequency. In the evaluation section, we'll show the relationship between the threshold value and the overall average number of memory accesses due to the query and to the movements triggered by the threshold condition.

An important consideration is that while the standard query operation has a fixed worst-case memory access time of \( d \) memory accesses, the conditional query has a higher and not constant worst-case access time, due to the triggering of the reallocation. If a constant worst-case access time is needed, the reallocation scheme can be triggered only at specific time intervals, using the standard query operation during the search phase, and a scrub operation as described before at specific periods to perform a global reallocation of the most-used items. As we’ll see in the next section, both the methods have similar performance in terms of average memory accesses. In the evaluation, all the queries are conditional when no scrubbing is used and normal when scrubbing is used.

It’s also worth noticing that many traffic monitoring applications implemented with Cuckoo hashing already implement the required basic functionalities needed to perform the reallocation, such as the need of a counter \( c_r \) and of a scrubbing procedure. For example, counting the number of query/update accesses is already done when the Cuckoo hashing is used to count the numbers of packets belonging to a flow. Therefore, in those flow monitoring applications, the proposed Cuckoo cache scheme doesn’t require additional bits per entry or writes, as the counter is already there. For other applications, the cost of maintaining a counter per entry can be significant. Also, scrubbing operations are usually scheduled in many traffic monitoring applications to remove inactive entries and also can be used to move frequently used entries with low additional overhead.

While the effects of the algorithm on a system with a plain memory hierarchy are clear, it’s worthwhile to describe the algorithm’s effect in a modern CPU, with a memory hierarchy that has multiple cache levels. Suppose that a frequently used item \( x \) is stored in table \( h_1(x), h_2(x), h_3(x) \) before retrieving the value \( v_x \) stored in \( h_4(x) \). Therefore, we argue that each frequently used item stored in the cache will require on average 2.5 cache slots. On the other hand, the proposed algorithm (by placing the frequently used items on the first table) will require only one cache slot per frequently used item. Therefore, the proposed algorithm would be able to store in the cache up to 2.5 times the number of frequently used items that are stored by the standard algorithm. Because a cache line will contain several items, there’s another benefit of the proposed scheme. Cuckoo cache concentrates the most used items on table one so that when caching a line, it’s likely that the other items are also frequently used items. This doesn’t occur in the traditional implementation, on which there’s no correlation between location and use of an item. The experimental data provided in the “Evaluation” section will confirm these intuitions.

Finally, if we also consider a CPU able to commit multiple memory requests in parallel to fully exploit the memory bandwidth and hide the memory latency, we believe that our proposed algorithm can provide a benefit with respect to the standard implementation. In fact, the available memory bandwidth can be used to perform multiple queries related to different packets in parallel (this is the trend in many packet-processing frameworks, such as PFQ\textsuperscript{20} and other works\textsuperscript{16}) instead of performing multiple memory accesses related to the same packet. This is well-suited to the proposed scheme that can complete many queries on the first memory access. The study of parallel accesses to the memory is left for future work.

Alternative Schemes

Now, we briefly discuss two alternative methods that can be used to reduce the number of memory accesses in Cuckoo hashing. These methods will be used to understand the proposed Cuckoo cache scheme’s effectiveness.

Cuckoo Affinity Hashing

The Cuckoo affinity hashing\textsuperscript{12} scheme modifies traditional implementations by using an additional hash function \( h_d(x) \). This function is used to set an affinity of each element to one of the \( d \) hash tables. Then insertion and query operations start with that table instead...
Measuring the Internet

...of always starting from the first table. For low and medium memory occupancies, affinity increases the probability that an element is found in the first memory access.

Optimal Cuckoo Hash Using a Chained Hash Table

The method we present here is able to put the best values (that is, the ones that have the maximum values of \( c_x \)) in the first table. This scheme can be useful for a static database but it’s difficult to use it in real applications. In any case, it’s useful for comparison because it gives the lower bound that can be achievable by exploiting the frequency information provided by the counters \( c_x \).

The method works as follows:

1) The set of items to store in the Cuckoo hash are first inserted in a hash table with separate chaining. This hash table uses \( h_1(x) \) (the hash function of the first table) as the hash function.
2) Order all the chains by value \( cx \). In this way, the first element of each chain is the \( x \) element with the maximum \( c_x \).
3) Put all the first values in the first table of the Cuckoo hash and remove from the hash table with separate chaining.
4) Insert the rest on items performing a Cuckoo insertion, but using only the \( d - 1 \) tables different from the first one.

At the end of this procedure, we have a Cuckoo table in which the elements in the first table are the ones with the best values of \( c_x \). It’s worth noting that, while this construction using the hash table with separate chaining provides optimal results, this isn’t true for the Cuckoo cache. An example can help to understand this effect. Suppose that there are three items \( x, y, z \) with \( c_z < c_y < c_x \), with the same \( h1 \) values \( h_1(x) = h_1(y) = h_1(z) \) and different \( h2 \) values \( h_2(x) \neq h_2(y) = h_2(z) \). Before applying the scrubbing procedure, \( x \) is in the first table, \( y \) in the second table, and \( z \) in the third table. First, the scrubbing procedure reaches \( y \) in position \( h_2(y) \) and moves \( y \) in the first table; it also tries to insert \( x \) in any table except the first one. Due to the cuckoo movements, it’s possible that \( z \) goes in the second table, but in a position \( h_2(z) < h_2(y) \) that has been already visited by the scrubbing procedure. Therefore, at the end of the scrubbing procedure, the item stored in the first table isn’t the optimal one \( z \), but the item \( y \). While this example shows that the Cuckoo cache scheme isn’t optimal, it also shows that the combination of events that insert in the first table a not-optimal value is unlikely to occur. Therefore, it’s expected that the difference between the performance achievable with the optimal construction and the one with reallocation will be small. Next, we show by means of experiments that this intuition is confirmed.

Evaluation

Let’s evaluate the average number of accesses of a Cuckoo table used to perform traffic monitoring of a set of hosts in a network. The application counts the number of packets exchanged by each pair of hosts. The simulations have been performed using a C implementation of Cuckoo hashing. For each occupancy factor, the simulation has been done counting the number of packets for each source IP host for a trace of one million packets. The traces used for evaluation are anonymized Internet traces taken from the University of California, San Diego’s Center for Applied Internet Data Analysis (CAIDA), collected 19 January 2012 (see www.caida.org/data/realt ime/passive). The number of different source IP hosts is 59,309 and the heaviest 25 percent of IPs account for 91 percent of packets. Two configurations of the application have been tested. In the first one, the application performs a conditional update on the Cuckoo hash for each received packet. The number of memory accesses due to the query and to the moving operations triggered by the conditional query are logged to help provide insight into the reallocation method’s behavior with the conditional update. In the second configuration, two traces are used. The first trace populates the hash table, after the scrubbing procedure is called, globally reallocating the most-used items, and then a standard query for each received packet of the second trace is performed. To better understand the performance of this method, the same simulations are performed with the two alternative schemes presented in the “Alternative Schemes” section.

The simulations have been done for a traditional Cuckoo hash with \( d = 4 \), with \( b = 1 \) elements per position and varying the size of the tables — that is, \{29,500, 24,500, 21,000, 18,500, 16,400, 15,600\} corresponding to an occupancy...
Cuckoo Cache: A Technique to Improve Flow Monitoring Throughput

The maximum table occupancy that can be achieved in Cuckoo hashing depends on the number of tables ($d$) and the number of elements per position ($b$). Four tables are used in many practical implementations (for example, in Pat Bosshart and his colleagues’ work), as it’s the lowest value that achieves close to 100 percent occupancy when $b = 1$. This was observed in the simulations, as the maximum occupancy for $d = 4$ was 97.6 percent.

As discussed previously, using values of $b$ larger than 1 increases the memory bandwidth that acts as the biggest bottleneck in an FPGA-based implementation that uses external memory.

Figure 1 presents the results in terms of the average number of memory accesses for each query for $d = 4$. This compares the standard, along with the affinity and optimal caching implementations.

The average number of accesses increases with occupancy, reaching values of approximately 2.5 as the load approaches 100 percent. Affinity reduces the number of accesses significantly when table occupancy is below 90 percent, but its benefits disappear as occupancy approaches 100 percent. Finally, optimal caching provides large benefits at all occupancies and clearly outperforms the other two methods. However, the optimal caching fails to achieve 95 percent occupancy. This is because although the first table can be almost full, it leaves many elements to be inserted on a Cuckoo hash with $d = 3$ on the remaining tables, and that achieves only an occupancy of approximately 90 percent. These results show the potential of caching. The problem is that in a dynamic scenario, optimal caching isn’t feasible and any caching scheme requires additional memory accesses to move the most-used items.

To evaluate the benefits of a practical implementation, we show the results for Cuckoo cache with scrubbing in Figure 2 and with conditional queries in Figure 3. In both cases, for different values of $th$, it can be observed that when the load approaches 100 percent, using small values of the threshold $th$ causes poor performance.
of $th$ causes poor performance. This can be explained as the cost of insertion increases with occupancy and the use of a small threshold causes frequent insertions that consume many memory accesses. On the other hand, the use of large values of $th$ degrades performance at low occupancies. This behavior is similar for both the scrub and the conditional query implementations. Therefore, it seems that a value of $th$ that’s a function of the occupancy should be used to achieve good results. The study of such an adaptive threshold is left for future work, as the focus of this article is to present the idea of a Cuckoo cache. Finally, both implementations require a significant overhead compared to the optimal caching scheme. A more detailed breakdown of the results shows that this overhead is mostly related to the memory accesses needed to perform the scrub or the movements in the conditional queries – that is, to place the most-used items on the first table.

To understand the performance of our algorithm when executed on a modern CPU, we execute the aforementioned test on a Linux workstation equipped with a single X5650 Intel Xeon CPU with 6 cores running at 2.67 GHz, with a 32-Kbyte level 1 (L1) cache, a 256-Kbyte L2 cache, and a 12-Mbyte last-level cache (LLC). In the experiments, the entire trace is first loaded on the memory and then packet headers are processed sequentially. The execution times reported are only for the packet processing and don’t account for the time required to load the trace from the disk. The entire Cuckoo tables fit on the LLC and therefore performance is bounded by the number of accesses to the LLC. The code used for the experiments is single-thread, so actually only one core is used to perform the experiments. Because the bottleneck in the L2 misses, we argue that a multithread application should have a similar behavior. Table 1 shows (for different occupancy rates) the execution times and number of accesses to the LLC for the standard query, the Cuckoo cache with scrubbing, and the Cuckoo cache with conditional queries – all with a value of $th = 50$. As expected, the proposed algorithms are faster than the standard algorithm. The scrubbing option provides the best execution times, especially for high occupancies. This is probably because insertions triggered by conditional queries require more memory accesses as occupancy increases. The number of LLC accesses is correlated with the execution time, as expected.

We introduced Cuckoo cache, an improvement to Cuckoo hashing for increasing the throughput of flow monitoring. The idea exploits the uneven distribution of Internet traffic to reduce the average query time of Cuckoo hashing, moving to the first accessed table the most-used items. Our evaluation shows that this approach can significantly reduce the average query time compared to Cuckoo affinity hashing, especially when the memory occupancy is close to 100 percent. Therefore, Cuckoo cache can be useful when implementing flow monitoring on high-speed links.

For future work, we intend to study parallel accesses to the memory, and explore an adaptive threshold that could yield even better results.

Acknowledgments

This work is partially supported by the EU Commission in the frame of the Horizon 2020 ICT-05 BEhavioural-BAsed (BEBA) forwarding project, grant 644122; and by the Spanish Ministry of Economy and Competitiveness in the frame of the excellence network Elastic Networks TEC2015-71932-REDT.
Cuckoo Cache: A Technique to Improve Flow Monitoring Throughput

References


Salvatore Pontarelli is a researcher with the Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT) at the research unit of University of Rome “Tor Vergata,” Italy. His research interests include fault-tolerant systems, hardware for high-speed networks, reliability of post-CMOS devices, and circuit design using unconventional devices. Pontarelli has a PhD in microelectronics and telecommunications from the University of Rome “Tor Vergata.” Contact him at pontarelli@ing.uniroma2.it.

Pedro Reviriego is an associate professor with the Electronic Design and Space Technology group at the Universidad Antonio de Nebrija, Spain. His research interests include fault-tolerant systems, performance evaluation of communication networks, and the design of physical layer communication devices. Reviriego has a PhD in telecommunications engineering from the Technical University of Madrid. Contact him at previrie@nebrija.es.
Video streaming, one of the most traffic-consuming Internet applications, will account for 80 percent of all consumer Internet traffic in 2019. Its great popularity and broad market prospects have attracted tremendous providers, including YouTube, Netflix, and Hulu. To provide a common and standard way to deliver video content over Internet, the Internet Engineering Task Force (IETF) first published the Real-time Transport Protocol (RTP) as a specialized streaming protocol in 1996.

A strong trend has emerged in recent years where we use HTTP everywhere, even for video streaming. As bandwidth continuously increases and HTTP-friendly infrastructure such as content delivery networks (CDNs) become popular, HTTP streaming is replacing the well-defined RTP to serve as the dominant streaming delivery protocol. Leading players such as Microsoft (www.microsoft.com/silverlight/smoothstreaming), Apple (https://developer.apple.com/streaming), and Adobe (www.adobe.com/products/hds-dynamic-streaming.html) all launched their own platforms and protocols toward HTTP streaming. In 2011, the Moving Picture Expert Group (MPEG) finalized the standardization on Dynamic Adaptive Streaming over HTTP (DASH), which is gradually employed by popular service providers like Hulu. The wide support of MPEG-DASH underscores just how necessary standardization efforts on underlying streaming delivery really are.

However, diverse upper-layer services for video streaming—including video skimming and bullet screens—still lack related standardization efforts. It’s estimated that it would take a viewer more than 5 million years to go through all the Internet videos generated in a month in 2019. Determining how to resolve the conflicts between limited watching time and massive video content has been a critical problem for the development of video streaming. The research community has proposed video summarization techniques to browse and manage video resources quickly and efficiently.

One of the leading video service providers in China, iQiYi (www.iqiyi.com), offers a video skimming service called LVjing. Different from previous skimming techniques that rely on video content analysis to select representative content, LVjing generates video skims based on many end users’ manipulation data on a seek bar (by determining, for example, how many viewers skip a specific period).

After witnessing a multitude of efforts from both research and industry, we believe video skimming is significant enough to be adopted as a standard service.

KaaS: A Standard Framework Proposal on Video Skimming

Lanshan Zhang, Linhui Sun, Wendong Wang, and Ye Tian • Beijing University of Posts and Telecommunications, China

Video skimming, a technique for video summarization, is particularly significant in the face of explosive growth of video content. This article highlights the necessities and motivations of providing video skimming as a standard service and proposes a standard framework for video skimming.
KaaS: A Standard Framework Proposal on Video Skimming

standard upper-layer service for video streaming applications. Thus, here we discuss the motivations for adopting skimming as a standard service, and present skimming-as-a-Service (KaaS), our proposed standard framework for video skimming. In this article, we explain our implementation over an open source streaming client and consider further deployment considerations.

How Did We Get Here?
To counter the explosive growth of videos, researchers have put forward the concept of video summarization and proposed various mechanisms. With the purpose of quickly browsing video content and efficiently managing video resources, video summarization focuses on presenting partial but representative sections of the full-length video to enable viewers to get a general idea in a short time. An effective and lightweight approach to video summarization is referred to as keyframes, which selects a set of representative and important static frames of the video to express the video summary.

Video skimming, typically considered another significant technique for video summarization, lets end users dynamically browse full-length videos in a limited time. Video skimming extracts important segments and reassembles them into a short video clip. Compared to its static counterpart keyframes, video skimming has advantages in expressiveness and impressiveness, because it remains in the form of video, but is more complicated and requires a better interpretation of video content.7

However, in the era of information explosion, video streaming is in the midst of a critical bottleneck. There’s an overwhelming discrepancy between the amount of full-length video and user-desired content available in relation to the amount of time users have for viewing and digesting such information. Eyeball time — a fresh concept that helps explain the problem — denotes how much time the viewer prefers or is able to spend watching a video. This amount of time varies widely among different viewers for different videos. The gap between eyeball time and a video’s actual length greatly restricts the user’s experience. Because video skimming’s major task is to select and present partial (but important) video segments, we believe that adopting skimming as a service could diminish the gap.

Note that this isn’t equivalent to skimming towards video summarization, because a video summary’s primary objective is browsing in a short time. Although it has been extended to adapt to user interactions,8 it’s hard to satisfy the diverse eyeball time of different viewers. As a service, the main goal of video skimming alters to offer a flexible playback pattern to reduce the gap between different viewers’ eyeball time and the video’s length.

With standard underlying HTTP streaming delivery provided, a standard framework for skimming service is also required to promote and benefit the actual service deployment. This is because it provides a common interface for different skimming-generation techniques or manners and splits deployment into two viable phases — video description and skimming playback. Besides, asking everyone to employ a standard framework makes the same skimming available on different platforms without duplicated efforts. Thus, we propose KaaS as a common and standard framework to explore skimming as a service.

KaaS’s Design
Now that we have a better sense of the history of video skimming, along with the motivation for our work, let’s look at KaaS’s design. We consider this a tentative standard framework for video skimming. To the best of our knowledge, KaaS is the first framework that aims to enable skimming as a standard service.

Design Principles
The primary objectives of KaaS are two-fold: first, to offer a common and standard presentation of video segmentation and description; and second, to enable lightweight but smooth video skimming based on the common presentation. To fully realize these objectives, the framework design of KaaS must comply with the following design principles.

Adaptability. Recall that eyeball time varies widely among different viewers for different videos. Referring to the same video, different viewers also have diverse eyeball time and content preferences. As a result, hundreds of skimming playbacks exist for a video that can be hardly realized by a single skimming technique. Meanwhile, even for the same viewer, the skimming playbacks for different videos (especially from different video domains) aren’t similar. We conclude such phenomena as the diversity of video skimming and the framework must adapt to the diversity consequently.

Flexibility. Besides diversity, video skimming also suffers its dynamic nature. It’s natural that the viewer’s watching habit isn’t fixed during a skimming playback. For instance, starting with a short skimming playback, the viewer will tend to switch to a longer skimming playback with more details if he’s attracted by the content. Hence, the framework must provide the flexibility to tackle the dynamic nature of video skimming that permits flexible switching among different skimming playbacks at any time without having to restart.

Compatibility. The framework must be compatible with the existing HTTP-based media delivery infrastructure and standard HTTP streaming protocol (MPEG-DASH). Ideally, the framework doesn’t modify the server side (such as HTTP origin servers and CDN servers) or change the MPEG-DASH specifications, and requires less modification to the client side.
KaaS’s Framework Design

Following design principles, we’ve come up with the framework design of KaaS. Figure 1 shows the high-level overview of the framework with a typical MPEG-DASH client. The section enclosed by a dotted line represents the core skimming system. It serves as a critical module, which associates with three traditional key components – the Media Player, Media Presentation Description (MPD) Parser, and HTTP client – of the DASH client to handle streaming sessions. Within the system, there are also three major components working closely to realize a lightweight and flexible video skimming. Before highlighting each component’s role, we first introduce a key standard element that provides the common presentation of video segmentation and description, meanwhile, supports the adaptability.

Multi-attribute Media Presentation Description (MMPD). Segmentation is typically regarded as the first step of video skimming generation. Various skimming-generation techniques usually have different segmentation manners, and without a common and standard interface, it’s hard to serve skimming as a general service. Hence, we introduce the MMPD, which describes those segmentations in a multi-attribute manner. Prior to the MMPD process, the video is always split into quite a few fine-grained segments through one or more skimming-generation techniques or merely a manual interpretation. The MMPD records and describes each segment with multiple attributes (keywords) summarized from one or more techniques or manual interpretation to depict its content.

Unlike Media Presentation Description (MPD) in MPEG-DASH, which concentrates on portraying the timing and other basic parameters (such as bitrates, URLs, and so on), MMPD focuses on the content and skimming perspective, which satisfies the following features. First, the presentation consists of a series of segments that don’t overlap in the time domain. Second, each segment has one or more attributes to describe its content. Third, each segment has indexing metadata (for example, the start and end times).

To enable skimming as a service, we also need user interactions (such as the content preference, eyeball time, and so on) from the front end. For instance, the framework selects a set of desired segments whose attributes are matched with the content preference and forms a video clip whose length also satisfies the eyeball time. Next, we highlight how the skimming system behaves to provide a skimming playback.
**MMPD Parser.** The MMPD Parser first retrieves an MMPD file through HTTP delivery and parses it to return a sequence of segment descriptions. Then the client acquires knowledge about each segment’s multiple attributes. To support the adaptability and flexibility, the MMPD Parser also stores various sequences of segment descriptions derived from different MMPD files.

**Segment Controller.** The Segment Controller monitors user interactions from front end and output from the MMPD Parser. Any valid inputs or changes trigger it to generate a latest user-desired sequence of segments. By tracking real-time playback status, it tells the Skimming Controller which segment should be played next. The “segment” discussed here isn’t an actual video segment but a virtual period with start and end times.

**Skimming Controller.** The Skimming Controller connects all the related components together to fulfill the eventual video skimming. It checks the current playing time and judges whether the upcoming time belongs to the latest user-desired sequence of segments. If not, it will simply skip to the target time and request the related video segment if necessary.

**Implementation and Example**

Here, we provide implementation details, along with a basic example of using skimming as a service through KaaS.

**Implementation**

To evaluate the feasibility and potential deployment of the proposed skimming service, we implemented the skimming system of KaaS on the master branch for dash.js (version 1.5.0), which is a standard-based MPEG-DASH client developed by the DASH Industry Forum.

**Selective introduction of dash.js.**

Figure 2 shows a partial architecture of the dash.js player, which concerns the streaming session playback. Serving as the core scheduling rule for playback, the PlaybackTimeRule controls the player to retrieve segments according to current state. We summarize the detailed process as follows. The ScheduleController periodically runs a validate function to call the PlaybackTimeRule. Then it will obtain the current bitrate and buffer state from AbrController and BufferController, respectively. With the information, it will retrieve segment through the DashAdapter with the help of an MPD file (parsed by ManifestLoader) to realize the playback.

**Our implementations and modifications.** To implement the skimming system within the dash.js player, we modified and extended the PlaybackTimeRule in two aspects: first, we replaced the next regular playback time with the time required from the skimming playback; second, we monitored the buffer state and had the buffer request segments on-demand. Such modifications enable the original PlaybackTimeRule to serve as the Skimming Controller. The MMPD Parser and Segment Controller are implemented as individual modules (see Figure 2).

**Example**

We developed a sample webpage of the dash.js player with the skimming system of KaaS implemented to illustrate a simple skimming service. In this case, a four-minute sample video is split into eight consecutive periods by parsing
Figure 4. A sample webpage of the dash.js player with the skimming system of KaaS implemented. Here, the attribute “desert” is selected as a content preference, with relevant periods marked as green dots, and irrelevant periods marked as red dots in the video player’s seek bar.

the MMPD file shown in Figure 3. Figure 4 illustrates the front end of this sample webpage demo; we listed all the attributes that describe those eight periods in the “User Preference” column for viewers to select. Here, we select the attribute “desert” as a content preference, and the relevant periods are marked as green dots while the irrelevant periods are marked as red dots on the video player’s seek bar. When the video is playing, the player will simply skip all the red periods to enable the desired skimming service.

The example illustrated here is the simplest (but also the most typical) use case. Consistent with the same basic principle of offering skimming service, further complicated use cases could provide viewers with diverse and personalized preference choices, or even recommend content automatically through machine learning.

**Standardization and Deployment**

Although researchers have come up with tens of skimming-generation techniques for video summarization, we haven’t seen a widespread deployment of video skimming in practice yet. Major reasons include the diversity of skimming-generation techniques and limitations of the applicability (or purpose) of video summarization. Assuming that video content analysis is getting better with the development of artificial intelligence and machine learning, deployment is still a dilemma without a standard framework.

If we used KaaS as a standard framework, though, MMPD would be the common interface, splitting deployment into two separate phases: multi-attribute video description and skimming playback. Next, we discuss how to deploy the skimming service with the separation brought by the standard framework.

As multi-attribute video description is a content-related business, we have two candidates for deploying it. Service providers (such as YouTube or iQiyi), usually serving as content owners at the same time, tend to be the first candidate to offer the skimming as an upper-layer service above basic video delivery. With the trend of providing developers with APIs to manage video content and customize video playback elsewhere, there may exist third-party skimming service providers to become another candidate to offer a specialized skimming service.

Because the skimming system is in charge of skimming playback, where to deploy it really matters. In the design we present here, we deploy it on the client side due to the simplicity of implementation. To adopt a skimming service under this circumstance, the service provider must modify its player (client). An alternative is to deploy the skimming system as an intermediary, which requires no changes from either the client or server side. However, the framework ends up being more complicated, because it can’t control playback directly.

Considering the incremental deployment, choosing a third-party skimming service provider is reasonable at first. To ask little efforts from different streaming services or websites, deploying the skimming system as an intermediary initially makes sense. Leveraging a standard framework such as KaaS also achieves the interoperability that makes the same video skimming available on different platforms, which eases the service deployment. When viewers are getting accustomed to the skimming service, content owners might prefer handling the video description and modifying their players to enable more efficient video skimming.

As a well-defined video summarization technique, video skimming isn’t a recent topic – but today’s enormous amount of video content pushes it to serve as an important upper-layer service for video streaming applications. With the standardization of underlying HTTP streaming delivery provided, there are clear trends and obvious advantages to standardizing skimming as a service. We hope our KaaS proposal and discussion offer
a valuable vision for future work on this topic.

Acknowledgment
This work is supported by the National Natural Science Foundation of China (grants 61402045 and 61370197).

References

KaaS: A Standard Framework Proposal on Video Skimming

Lanshan Zhang is an associate professor in the Beijing Key Laboratory of Network System and Network Culture at Beijing University of Posts and Telecommunications, China. Her research interests include media streaming and digital imaging. Zhang has a PhD in broadcast and TV art from the Communication University of China. Contact her at zls326@sina.com.

Linhui Sun is a graduate student in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. Her research interests include network management and HTTP. Sun has a BS in telecommunications engineering with management from Beijing University of Posts and Telecommunications. Contact her at lh.sunlinh@gmail.com.

Wendong Wang is a full professor in the School of Software Engineering at Beijing University of Posts and Telecommunications. His research interests include future Internet architecture, software-defined networking technologies, and network or service QoS. Wang has an ME in computer science from Beijing University of Posts and Telecommunications. Contact him at wdwang@bupt.edu.cn.

Ye Tian is a postdoctoral researcher in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications. His research interests include future Internet architecture and data mining. Tian has a PhD in computer science from Beijing University of Posts and Telecommunications. Contact him at humanity@163.com.
What Is Algorithm Governance?

Danilo Doneda • Rio de Janeiro State University

Virgilio A.F. Almeida • Harvard University

With algorithms' increased use to fulfill complex tasks comes the risk of algorithms' use for manipulation, biases, censorship, social discrimination, violations of privacy and property rights, and more. To address such risks, the process of algorithm governance should be considered.

Algorithms are basically sets of instructions to perform a task, producing an output from a given input. Today, algorithms embedded into systems and electronic devices are increasingly trusted to make decisions, evaluations, and analysis with a concrete impact on our lives.

Algorithms' vocation to penetrate several realms of our routine is now regarded as a fact of life. They perform tasks we could hardly imagine accomplishing without a human in charge. As their sophistication and utility grows, the more they seem “autonomous” — even evoking the notions of a “thinking machine” present in some arcane thoughts that date back to the early age of computer science. In fact, often the term “algorithm” is being used or referred to as a synonym for a computer, machine, code, software, and so on.

The availability of ever-increasing computing power and datasets make it possible for algorithms to perform tasks with a magnitude and complexity that’s unbearable by human standards. Often their results can hardly be anticipated or explained — even by their designers.

At the same time, although they provide valuable output, algorithms can also take humans out of the loop in their series of decisional processes, which can be risky. Thus, to foster their integration into several social and economic processes in which they can be valuable, perhaps we should design instruments that permit some sort of governance of algorithms. This could prevent them from affecting the balance of power negatively, in favor of those who are able to exercise actual power regarding how they’re used, as well as maximizing their benefits and reducing their potential risk. An example of such a shift in the balance of power is given by Frank Pasquale when he mentions that a woman who had taken antidepressants as a sleep aid had her requests for health insurance denied by some companies. The record of her use of these medications, that could eventually help her if it was kept for strict medical usage, was used against her on the basis of presumptions regarding her use of these medicines.1

Algorithms' Potential Issues and Elements

The complexity of algorithms' work is increased by the growing use of machine-learning techniques. With these techniques, an algorithm can rearrange and morph itself and its inner workings based on the data it analyzes. As Pedro Domingos once described, learning algorithms are “algorithms that make other algorithms... so we don’t have to.”2 Often it isn’t a simple task for the data scientist or the algorithm’s designer to describe the steps an algorithm has taken to produce a certain output, if not in merely abstract terms.

Hence, algorithms add a new element to the information processing chain — their opacity — that often is associated with the difficulty of decoding their output. Humans are increasingly unable to understand, explain, or predict algorithms' inner workings, their biases, and eventual problems. This is cause for rising concern in situations where
algorithms are trusted to make important, if not fundamental, decisions that affect our lives, to the point that a call for more transparency and accountability of algorithms is being increasingly noticed in academic works as well as public campaigns.3

At the same time, there are also non-technical justifications for their opacity. Some of them are based on issues over competition. Having an open algorithm could put a company at a disadvantage regarding its competitors. Others are based on intellectual property: in some countries, the law protects a company’s commercial secret or intellectual property. Another reason for not opening certain algorithms is the possibility that some people — once they’re aware of their characteristics — could be able to better “game” the algorithm.4 So, the opacity of algorithms is a tendency sustained by elements of both technical and non-technical nature.

Opacity, though, hasn’t been a barrier to the widespread adoption of algorithms in several realms. In fact, algorithms are no longer seen as the trick behind search engines or something that helps e-commerce gather customers’ preferences — instead, they’re now essential parts of self-driving cars, crime-prediction frameworks, and tests for many diseases, along with a growing list of other important applications. And several of these applications have a direct impact on society, from their employment to make sense of data for development and humanitarian action (see www.unglobalpulse.org), to aiding doctors in finding the right diagnosis, or even enabling more rationality in judicial decisions.

Algorithms have risen to perform an ever-increasing number of tasks, due not merely to their own development per se, but to the occurrence of conditions that transformed the whole environment they’re settled in. Indeed, “the algorithm isn’t an algorithm because it executes the instructions; it’s an algorithm because it’s enacted as such by a heterogeneous assemblage of actors, imparting to it the very action we assume it to be doing.”5

This environment contains elements of major relevance to algorithm’s governance — in fact, their governance can often be based on tools that work not on the algorithm itself but on elements of their environment. Out of these elements, datasets are probably the most fundamental ones. Algorithms became much more useful as a function of the availability of data, which is relevant for its inner work. As Tarleton Gillespie notes, “Algorithms are inert, meaningless machines until paired with databases upon which to function.”6

Datasets are built up from the data that are collected in a fast-paced rhythm, as more and more of our activities leave a trace (think of our actions on the Internet) or are monitored. As a consequence, the amount of relevant data available grows dramatically. This is indeed central to the idea of Big Data, the paradigm for data that often “feeds” algorithms, with characteristics that are often referred as the 3 V’s: volume (more data are available), variety (from a wider number of sources), and velocity (at an increasing pace, even in real time).6

If datasets are used as central parts of the tasks performed by algorithms, it’s important to emphasize the need to verify if they’re being lawfully and even ethically used — in short, if the data are legitimate, correct, updated, and don’t show any type of bias. For instance, data mining and other methods used to refine the datasets can eventually result in discrimination. Furthermore, selection, classification, correlation, and other techniques many times tend to replicate environmental bias, as they can mimic social and personal conditions. This isn’t exactly a novelty, as statistical discrimination (the drawing of stereotypes based on the “average” behavior of a discriminated group) has been an issue for more than four decades, but this is a problem which algorithms are constantly making more salient.7

Governing Algorithms
Several potential risks of the use of algorithms have been identified in the literature, such as the risks of manipulation, biases, censorship, social discrimination, violations of privacy and property rights, abuse of market power, effects on cognitive capabilities, and growing heteronomy. To address these risks, the process of algorithm governance should be considered.

Algorithm governance can vary from the strict legal and regulatory viewpoints to a purely technical standpoint. Its focus is often on accountability, transparency, and technical assurances. The resource to a certain governance path can be based on factors such as the nature of the algorithm, its context, or risk analysis.8

Generally, when a governance option is made it aims to reduce problems caused by the algorithm. It should try to preserve its effectiveness and reduce undesirable outcomes.

Some governance tools act not on the algorithm but on the data they need in order to work. This is true for several tools already present in data-protection legislation that, in some countries, have measures regarding transparency and fairness that apply directly to algorithms and the platforms that support their functioning. For instance, the provision that automated decisions shall be grounded on transparent criteria is commonly present in several pieces of data-protection legislation. The same happens with the right to ask for a human revision of automatically taken decisions.

Using algorithms to regulate datasets is at the core of most data protection legal frameworks, which also command these datasets to be legitimate and rightful, undergoing several requirements to meet this criteria. An example would be consent for the use of personal data on various occasions, as ownership is another rising issue, and identifying specific datasets — in a manner to allow consent over the treatment and use of data, whether for personal use or just
Internet Governance

originated by a citizen — should also be an issue for regulation.

The need for accountability and transparency of algorithms is often mentioned as another possible approach. Transparency, as we’ve already mentioned, isn’t natural to many algorithms in use, for technical and non-technical reasons, so we need governance instruments to foster the adoption of certain transparency levels or open algorithms.

Accountability, which is linked to the notion of responsibility, fairness, and due process on the use of algorithms, is also fundamental and calls for another question that will be faced with the widespread use of algorithms: who’s liable for their use? In which situations will algorithm designers be deemed liable versus when an enterprise or a government body which employs the algorithm will?

Technical assurances are another fundamental resource, to provide the design options for data mining and analytics with considerations that aim to evade prejudice, inequality, or other biased outcomes. In this realm, engineers and researchers are developing techniques for guaranteeing that algorithms and their implementations shall satisfy design, performance, and even liability standards. In a further step, there are auditing techniques that can be useful to determine if the algorithm meets the technical required standards.

A tool closely related to self-regulation is the development of principles regarding the ethical use of personal data — which is being mentioned sometimes as Big Data Ethics. Even if it’s a variation of the self-regulatory approach, some governmental bodies have mentioned that perhaps these principles should be developed into a new regulatory framework.

Another important element is that algorithms are constantly working on the fly, facing new and unprecedented situations that require answers, thereby necessitating our constant monitoring of their outcome for evaluation. This is even more important in the case of machine-learning techniques.

Implementing governance instruments for algorithms can occur at multiple levels. Here, we describe some of these levels, taking into account that some of them would only be considered if the risk that certain algorithms present is substantial and concrete. Algorithm governance processes can range from market-oriented solutions to government-based mechanisms.

An ensemble of oversight bodies is required for structuring and implementing algorithm governance on a variety of instruments. It’s evident that there’s no one-size-fits-all solution.

Private companies should approach the use of algorithms with given standards (if their customers are in a position to refrain from using risky algorithms built into their software, services, and products), as long as there are adequate levels of transparency and accountability in place.

For this private sector approach to work systematically, it should be built on a company’s internal organization, where it defines standards that reflect public interest and establishes a reviewing process and an internal body to guarantee the integrity and compliance with public interest values when using algorithms.

It can also rely on industry-wide self-regulation processes where, for instance, collective standards and public interest values are defined for a specific sector — as happens, for example, with the auto industry defining quality and security standards for car-embarked software. A specific industry oversight body, which can take the form of a multistakeholder committee, would be responsible for demanding information from software makers about algorithms.

Finally, a governmental oversight body in charge of algorithm regulation is another possibility for the future, focusing on requirements such as the level of transparency or quality of service in terms of errors, risks of death, or injuries caused by algorithms or software, along with security breaches and other concerns.

Acknowledgment
We thank Yasodara Córdova for her valuable insights and suggestions.

References

Danilo Doneda is a professor of civil law at the Law School of the Rio de Janeiro State University (UERJ). His research interests include private law and regulation, privacy, and data protection. Doneda has a
What Is Algorithm Governance?

Virgilio A.F. Almeida is a professor in the Computer Science Department at the Federal University of Minas Gerais (UFMG), Brazil, and currently a visiting professor at Harvard University. His research interests include large-scale distributed systems, the Internet, social computing, and cyber policies. Almeida has a PhD in computer science from Vanderbilt University. He’s the chairman of the Brazilian Internet Steering Committee (CGI.br). Contact him at virgilio@dcc.ufmg.br or valmeida@cyber.law.harvard.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

IEEE Software offers pioneering ideas, expert analyses, and thoughtful insights for software professionals who need to keep up with rapid technology change. It’s the authority on translating software theory into practice.

www.computer.org/software/subscribe
From the Service-Oriented Architecture to the Web API Economy

Wei Tan • IBM T.J. Watson Research Center
Yushun Fan • Tsinghua University, China
Ahmed Ghoneim and M. Anwar Hossain • King Saud University, Saudi Arabia
Schahram Dustdar • TU Wien, Austria

As Web APIs become the backbone of Web, cloud, mobile, and machine learning applications, the services computing community will need to expand and embrace opportunities and challenges from these domains.

Service-oriented architecture (SOA) made its debut in the early 2000s as a new architecture pattern. In this pattern, software components are encapsulated as individual services (or Web APIs), and invoked from the network through standard Web protocols such as HTTP. Web APIs are lightweight alternatives to WSDL/SOAP-based services that usually use REST as the communication protocol and JSON as the content format. A service usually represents a minimal reusable component that can be combined with many other services, forming value-added business processes, also known as composite services. The SOA paradigm with the accompanying Web service protocols – including SOAP, REST, Web Service Definition Language (WSDL), and Web Services Business Process Execution Language (WS-BPEL) – has become the de facto standard in enterprise information systems to achieve interoperability.

These days, SOA and services computing have gone much beyond interoperation technology (see Figure 1). REST-style Web APIs have replaced SOAP services for two reasons: first, REST’s create, read, update, and delete (CRUD) interface greatly improves consumability; second, JSON with REST makes the communication payload much simpler and easier to understand, compared to XML with SOAP. As evidence, starting in 2006 Google abandoned SOAP and only uses REST in its search APIs. While SOAP/WSDL is still popular in many enterprise systems, REST-style Web APIs are pervasive in Web, mobile, cloud infrastructure, and Internet of Things (IoT) applications. According to ProgrammableWeb (http://programmableWeb.com), the largest online API registry, Web API enjoyed a compounded annual growth rate of 100 percent (approximately) from 2005 to 2011, in terms of the total number of APIs registered. As of March 2016, ProgrammableWeb has listed more than 14,700 APIs. With the formation of this API ecosystem, an API economy is emerging. To give you a few examples, 60 percent of Salesforce’s transactions go through its APIs instead of the traditional Web GUI, contributing to its 1.3 billion daily transactions and more than $5 billion in annual revenue. Additionally, 90 percent of Expedia, 60 percent of eBay, and 100 percent of Amazon Web Services (AWS) revenue are from APIs.

Emerging Application Domains of Web APIs

Web services, particularly Web APIs, are becoming the backbone of Web, cloud, mobile, and machine learning applications. Thus, we argue that the services computing community should extend its scope, and embrace the newly emerged opportunities and challenges from these domains.

Web Applications

Web application developers can create a service composition (that is, a mashup), by combining multiple services. For example, we can create a trip itinerary service by combining a map service with a flight, a train, a rental car, and a hotel...
booking service. ProgrammableWeb has listed more than 6,000 mash-ups (www.programmableWeb.com/category/all/mashups).

Cloud Applications
Infrastructure-as-a-service cloud services, such as computing services that provide virtual machines, storage services that provide block or object storage, and message services that provide reliable message queues, are becoming the “utility providers” of many Internet businesses. As an example, Netflix uses various services from AWS, including virtual machines, storage, message queues, and databases. As another example, Dropbox (which offers online file storage and synchronization services) stores all its customer files on Amazon Simple Storage Service (S3).

Mobile and IoT Services
Nowadays, the so-called IoT — including smartphones, vehicles, wearables, smart home appliances, and smart factory machines — are connected to the cloud and the Web, or interconnected with one another. Studies have shown that many Web APIs such as advertising, social network, messaging, and billing are commonly used in mobile apps (see Figure 2).

Machine Learning and Big Data Services
Machine learning services are becoming important in the new wave of AI hype. Business owners want to focus on their core competence and source some non-essential but still very important features to a third party who possesses a given expertise. This isn’t a new phenomenon, but it’s becoming particularly interesting in the Big Data era. First, many companies (such as electronic commerce and content streaming) need machine learning capabilities, including image recognition, natural language processing, and recommendation as an integral part of their business. However, they usually don’t own the sophisticated machine learning algorithm, the sufficient training data to deliver a decent model, or a computer system capable of handling a big volume of training data to derive a model in a timely manner. This is a sweet spot for services computing — occupying the niche where a business wants to use a certain capability, but others are in a better position to deliver that.

Currently, two categories exist for machine learning services. The first category is a platform service through which users provide their own training data and receive a trained model. After training with customer data (training can take much time, depending on the data size and model complexity), the derived model can either be retrieved or stay in the same cloud to serve the inference purpose. Services of this category include Amazon Machine Learning (https://aws.amazon.com/machine-learning), Microsoft Azure Machine Learning (https://azure.microsoft.com/en-us/services/machine-learning) and Google Prediction API (https://cloud.google.com/prediction). An exemplary use case of that is to upload many emails marked as spam or non-spam, to train a classification model that acts as a spam filter for future emails.

The second category is a software service where users don’t provide any training data. Instead, users only provide the data to be inferred. Services of this category include Google Translate API (https://cloud.google.com/translate) and IBM Watson Developer Cloud (www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud). An exemplary use case of that is to upload a sentence in Chinese and get its translation in English, or to upload a photo and let the API recognize the objects in it.

In the following, we discuss two case studies that further explore the machine learning and IoT categories.

Figure 1. A brief history of services computing. After debuting in the early 2000s, services computing has transitioned from an enterprise interoperation technology to shape the Web API economy. (AWS = Amazon Web Services; BPEL4WS = Business Process Execution Language for Web Services; SOA = service-oriented architecture; WSDL = Web Services Description Language.)
Case study 1. In this scenario, the focus is on a faster and cheaper recommendation service using GPU acceleration. Recommendation is a key technology for online merchant and content-streaming companies. As an example, 80 percent of Netflix's watching hours are influenced by its recommender system. However, many of the recommender systems either require a sizable infrastructure or perform slowly when the number of users and items grows. For instance, Chao Liu and his colleagues mentioned starting their recommendation model training on Friday afternoon and getting the result by Monday morning. It follows, then, that an e-commerce website similar to Amazon or a digital content streaming provider similar to Netflix could only update the recommendation model once a week. This workaround isn’t viable when business becomes global and available every day around the clock.

Our work proposes using GPUs to accelerate (collaborative filtering-based) recommendation. By exploiting the massive parallelism inside individual GPU devices and across multiple devices, a matrix factorization tool called cuMF (https://github.com/weitian/CuMF) is able to offer recommendation services up to 10 times as fast, and up to 100 times as cost-efficient, compared to state-of-the-art distributed CPU solutions. This work sheds light on how a service’s quality can greatly improve by adopting an advanced computing infrastructure.

Case study 2. This scenario focuses on connecting people, mobile phones, and smart IoT devices. With more Internet-accessible IoT devices, connecting them via Web services with mobile phones and people is often the best way to utilize them. For example, someone might prefer to get an emergency call if his smart smoke detector raises an alarm; or perhaps that same person would like to receive a notification from a weather service if it predicts rain tomorrow, along with a reminder to take an umbrella and tell his smart irrigation device to delay watering the lawn. Mobile apps such as IFTTT (an abbreviation of “If This Then That;” see https://ifttt.com) let end users create chain APIs called “recipes,” which can connect Web services such as email, social network, online photo storage, and IoT devices (including smart home appliances, vehicles, and factory machines).

The Research Community’s Shifting Interest
Table 1 shows the major topics of interest to the services computing community, and their relation to other computer science areas.
community, and their relation to traditional computer science areas. Clearly, services computing is an interdisciplin ary area. It borrows methodologies and technologies from areas such as software engineering, the Web, databases, systems, AI, HCl, and operational research (OR) to tackle its specific problems.

To further gauge the community’s interests over time, we collected titles from IEEE International Conference on Web Services (ICWS) papers from 2015 and 2005, respectively. The word clouds in Figure 3 clearly illustrate a focus on data, mashup, and recommendation for 2015; and a focus on semantic, grid, and BPEL4WS for 2005.

In looking at the services computing community’s shift in interest over time, and determining the best paths moving forward, we offer two suggestions to researchers and practitioners:

1) Pay more attention to more practical approaches to solve real-life problems.
2) Pay more attention to applications in emerging areas such as mobile, IoT, and machine learning.

Regarding our first suggestion — let’s keep in mind that in the adolescent age of services computing, studies in this area cover some ivy-tower topics such as automated composition, verification of service processes using formal methods, and Semantic Web services. Researchers came up with sophisticated formalism and methods to tackle these problems, only to find that they aren’t close enough to reality. As an example, our earlier empirical study on scientific workflows shows that most service compositions use only a handful of services. As a result, the service processes are relatively simple. Therefore, in most real-life cases it would be an overkill to use formal methods such as Petri nets and process algebra to analyze them. At the same time, people have found that fully automated service composition is also far from reality. A more realistic approach is to provide context-aware recommendations in a composition’s design phase. For example, a recent study takes a highly innovative approach to facilitate developers and even end users accomplishing the composition in an interactive fashion.

Thus, when services computing moves from academia to industry shaping an API economy, people expect the technologies to be more practical and address the pain-point of developers.

Regarding our second suggestion, we argue that services or APIs shouldn’t limit themselves to enterprise integration. Instead, researchers and practitioners should pay attention to emerging areas, including mobile/IoT, network, Big Data, and machine learning services. Because these areas are quickly developing and evolving, and have a big demand for using service as the delivery channel, we believe that services computing researchers would find this area of development more valuable.

Acknowledgments
This work is supported by the National Natural Science Foundation of China (61174169). We also extend our sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this International Research Group (IRG14–28).

References


Wei Tan is a research staff member at the IBM T.J. Watson Research Center. His research interests include GPU acceleration, large-scale machine learning, service-oriented architecture, business and scientific workflows, and Petri nets. Tan has a PhD in automation engineering from Tsinghua University, China. Contact him at wtan@us.ibm.com.

Yushun Fan is a professor at Tsinghua University, China. His research interests include computer integrated manufacturing, service computing, intelligent service platforms, workflow, and Big Data. Fan has a PhD in automation engineering from Tsinghua University. Contact him at fanyus@tsinghua.edu.cn.

Ahmed Ghoneim is an assistant professor in the Department of Software Engineering, College of Computer Science and Information Sciences, King Saud University, Riyadh, Saudi Arabia. His research interests address software evolution; service-oriented engineering, software development methodologies, quality of service, net-centric computing, and human-computer interaction (HCI). Ghoneim has a PhD in software engineering from the University of Magdeburg, Germany. Contact him at ghoneim@ksu.edu.sa.

M. Anwar Hossain is an associate professor in the Department of Software Engineering, College of Computer Science and Information Sciences, King Saud University, Riyadh, Saudi Arabia. His research interests include multimedia surveillance and privacy, ambient intelligence, the Internet of Things, and cloud computing. Hossain has a PhD in electrical and computer engineering from the University of Ottawa, Canada. He’s a member of IEEE and ACM. Contact him at mahossain@ksu.edu.sa.

Schahram Dustdar is a full professor of computer science and he heads the Distributed Systems Group at TU Wien, Austria. His work focuses on distributed systems. Dustdar is an IEEE Fellow, a member of the Academia Europaea, an ACM Distinguished Scientist, and recipient of the IBM Faculty Award. Contact him at dustdar@dsg.tuwien.ac.at; dsg.tuwien.ac.at.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
Bot detection has become an important topic in security. Bots are software programs that use a computer, typically a personal computer, for malicious use or at least use unintended by its owner. So-called “aiming bots” were once quite popular in online multiplayer first-person shooter games. These bots allowed players to bypass the game mechanics for targeting opponents, giving them perfect aim every time, and enabling them to artificially improve their standing in the game. More commonly these days, bots are often employed to register for free email accounts and send spam or phishing messages. These, among many others, are scenarios that bot detection techniques are designed to avoid.

Two families of techniques for bot detection are in common use today. One is represented by CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) technology. The premise behind CAPTCHA is to require a human to interactively solve a problem that’s difficult (or more desirably, impossible) for a computer to solve. The now-ubiquitous CAPTCHA technology on the Internet involves having users look at distorted images of words or listen to distorted audio of words and type in the letters. Another approach, common in massively multi-user online games (MMOGs), involves monitoring a user’s input to identify characteristic differences between human and bot-like behavior. Steven Gianvecchio and his colleagues, for example, show that differences in the distributions of keystroke durations and the efficiency of mouse movement can be used to distinguish humans from bots. In online poker, systems can use this information to identify poker bots, along with other heuristic clues such as playing too many games continuously for too long a period of time. Yang-Wai Chow and his colleagues propose that CAPTCHAs can be integrated into a MMOG as a mini-game: for example, making progress in a fantasy adventure game might require players to decode spells, presented visually as CAPTCHAs. The result is more natural interaction, integrated into the context of the game, with players potentially enjoying the challenge and getting better with practice.

Both of these approaches require users to “prove” that they’re human; one requires explicit action on the user’s part, while the other is passive. In other words, one is a human interactive proof (HIP), the other a human observational proof (HOP). CAPTCHAs are a common examples of HIPs. An example of an observational proof is examining the spatial signature of mouse click locations, as influenced by an interface layout. In this article, we discuss HIPs and HOPs, along with other recent developments (namely, human subtlety proofs, or HSPs), and their potential for natural interaction. Many of our examples will come from online games, in part because of the enormous growth in popularity over the past decade or so, and in part because games support a variety of interactions that can be considered natural in a specific context.

Natural Interaction
What would natural techniques for bot detection look like? As a term of art in human–computer interaction, natural user interfaces have a few key properties. Following the account of Daniel Wigdor...
and Dennis Wixon,5 users enjoy interacting with a natural user interface; users become more skilled with practice; and their interaction is appropriate to context. Wigdor and Wixon summarize by saying that a natural user interface is one that makes a user act and feel like a natural.

We take naturalness to be an important part of interactive security techniques. Security measures tend to be viewed by the average computer user as overhead—not perhaps necessary, but still secondary in importance to carrying out other tasks.6 In the early days of computer security, Jerome Saltzer and Michael Schroeder7 identified psychological acceptability as an essential aspect of the human interface, making the correct use of protection mechanisms routine, but this goal has yet to be reached. To illustrate what we mean by “natural,” let’s discuss user authentication to establish some of the context for bot detection.

The use of usernames and passwords is a counterexample to natural interaction for user authentication. It’s long been understood that remembering a username and password depends on access to semantic information in long-term memory. Although typing can become automatic and natural, the retrieval of specific information from memory, with precise syntactic properties, has less of the same naturalness. To overcome this artificiality, users tend to rely on tricks that might be learned or practiced.

Other user authentication techniques, also in the challenge-response family, offer more naturalness. Biometric authentication, in which a user is identified by some physiological characteristic unique to that person, includes fingerprint analysis, face recognition, retina or iris scanning, speech and vocal sampling, gesture, hand geometry matching, and other techniques. A “natural” evolution of a technique such as fingerprint authentication is its integration into contexts in which users automatically place their fingertips in contact with the sensing device: a computer mouse, for example, or even the keys on a keyboard. Context plays an important role in the naturalness of the interaction. For example, Alexander Chan and his colleagues8 describe the use of a Leap Motion to capture hand geometry and gestures, with greater than 99 percent accuracy in authentication; such a system would be useful, providing even continuous authentication, in a context in which gestures are an integral part of interaction with the system.

A different avenue toward natural authentication is to rely not on fixed or slowly changing physiological characteristics of the user but on behaviors, which might be learned or practiced. These approaches tend to have a smaller footprint than those described above. The most common approach is to monitor a user’s mouse movement or keyboard actions, matching observed patterns against the user’s “signature.” Techniques typically involve automatically extracting or generating a large number of features from the time series of mouse movements or key presses, inducing a model from some subset of the features, and using that model to classify new observed data.4 Accuracy can be greater than 90 percent, with large enough samples of data, although acquiring enough data poses a tradeoff with respect to time.

The more natural interface techniques share one thing in common: they’re characteristically observational in nature. These approaches work in the background, collecting information about users to compare against a model, and require no explicit, intentional interaction from users—and therefore have little or no cost to users.

HIPs, HOPs, and HSPs

HIPs and HOPs both have significant limitations, however. HOPs are susceptible to imitation attacks, in which bots carry out scripted actions designed to look like human behavior. HIPs, on the other hand, tend to be more secure because they require explicit action from a user to complete a dynamically generated test. Because these tests are dynamically generated, solutions to them can’t (reasonably) be predicted, scripted, or generated by computer systems; however, because humans have to expend cognitive effort to pass HIPs, they can be disruptive or reduce productivity, violate the good design principles of natural interfaces, and even result in users seeking alternative systems to use.

With this in mind, some of the work happening at North Carolina State University is aimed at developing knowledge and techniques to enable human subtlety proofs. HSPs blend the stronger security characteristics of HIPs with the unobtrusiveness of HOPs. We’re examining how subtle cognitive biases affect interaction with software in predictable and repeatable ways. Our goal is to leverage those biases to make small changes to interfaces that will subtly—not substantively—affect the interaction of either bots or humans. By making changes to interfaces strategically and looking for evidence of the subtle changes that...
only humans would exhibit (and bots would find very difficult to fake), we expect HSPs to combine the strengths of both HIPs and HOPs.

One of our projects is based on touch interfaces on tablet computers. A sample interface presents a scattering of circular targets on the display; the user is to tap each target, upon which it disappears, to complete the task. Touch interfaces are associated with a much higher error rate in target selection than GUI interfaces used with a mouse or touchpad, and in practice a touch sometimes fails to register. Experimentally, with a tablet instrumented to collect touch and gaze data, we’ve identified different ways in which users respond to such errors.

One behavior involves a gaze fixation on a target, a tap, and then a pause to verify that the tap has been recognized. If the tap is successful the next target is handled, but if the tap fails then the target is tapped again. Another behavior is to tap targets without waiting for verification, returning to those that were missed. In either case, visual attention might remain on the target under consideration until a successful tap or move on to the next target. A different behavior, apparently derived from gaming experience, relies on peripheral vision to locate targets, with no obvious relationship between gaze fixations and tap locations. Yet another behavior involves a brief planning phase in which gaze moves between different targets before any one is tapped. These behaviors can be decomposed into what are referred to in the cognitive modeling literature as strategies (or micro-strategies) dimensions. Behaviors can be associated with different strategies by analysis of gaze fixations, tap locations, and the duration and ordering of events.

Critically, we can influence the target-selection error rate, either directly (by making targets larger or smaller) or indirectly (by simply ignoring taps, with some probability). The implication for HSPs is that if users are sensitive to the difference in error rates (we have evidence that this is the case), then we might be able to manipulate the interface to see if the user reacts in a way that we expect — for example, adopting a slower, more “careful” strategy for a higher error rate. Multiple target selection is a common enough task in touch-based interfaces that it could potentially act as the background for an HSP; this is part of our ongoing research.

In another project, we used a typing game as an experimental platform. In the game, players typed words with differing characteristics and of varying length: dictionary words, dictionary words with transposed letters, and words composed of random letters. The game rewarded players for typing as quickly and accurately as possible; one factor that varied in the game was whether players could retry a given round consisting of a certain type of words, without a penalty, to improve their performance. We found that typing speed improves with familiarity with words and with practice, but that these are independent of the number of mistakes that are made when typing. Specifically, the inter-keystroke interval (IKI), which is a measure of the time between key presses on a keyboard, was higher for misspelled words or random letter combinations. In all cases, the IKI decreased as players got practice typing those words or letter combinations. Figure 1 illustrates how practice reduces the IKI consistently for all word types, independent of players’ familiarity with what they’re typing. Figure 2 depicts the number of errors, which doesn’t significantly change across different word types.

Our interpretation of these results was the players were sensitive to the speed/accuracy tradeoff, depending on the cost of the typing errors they made, which has implications for security. The number of incorrect attempts is a commonly used security measure, but our data suggest that speed has a more direct relationship to the nature of what’s
Natural Web Interfaces

![Diagram showing the mean number of mistakes made by players for different word types.](image)

**Figure 2.** The mean number of mistakes made by players for different word types.

being typed than the number of mistakes that are made while typing. The implication of this finding is that by inspecting typing speed in conjunction with errors, a system can be more effective at detecting anomalies than looking at the number of incorrect attempts alone. Similarly, our results indicate that typing speed is correlated with the familiarity of the text being typed, which can be used, for example, to help systems ensure users create more secure passwords. It’s a common IT policy to ask users to create new passwords, but it’s impossible to know if a user’s new password is entirely new or has been used on another system somewhere. By comparing to a known baseline, such as the user typing their previous passwords, the subtleties of the manifestation of cognitive function on typing can be brought to bear to identify familiarity. While challenging the user to type a specific word is the most obvious and direct application of this work (an HIP task), the underlying reasons for user performance might be applicable in an HSP.

HSPs are a new technology under early development, but show promise in producing new avenues for creating more secure systems. Human perception, cognition, and motor systems govern the ways in which users interact with input devices, and those systems are heavily influenced by the task being performed and the interface used for the task.

For example, many users have the taskbar configured to auto-hide when the mouse isn’t in close proximity. When switching tasks while the taskbar is hidden, users are moving the mouse pointer to a target in their memory, rather than one identified by their visual perception. Under these different conditions, the motion of the mouse will differ in response to the brain’s different perceptual and cognitive functions. Identifying these differences bears a resemblance to standard HIPs; however, by showing the taskbar at different times, a system can influence those perceptual and cognitive functions in predictable ways to produce expected changes in mouse movements. Making this change to the task environment gives this approach more in common with standard HIPs. Combined, however, the unobtrusive observational nature of monitoring mouse movement with the subtle, but still on-task, change to the taskbar’s visibility will engage the subtlety of human cognition to produce an HSP.

Further down the line, similar approaches could yield techniques for detecting other classes of users, as well. Although bots and humans often have highly different interaction signatures, what about a user who’s very distracted and likely to make a mistake compared to a focused user? How about an expert who has familiarity with the task and interface in comparison to a novice? These distinctions might represent new avenues for securing systems from authorized — but unintended — uses.

**Acknowledgment**

This work was funded by the US National Security Agency and US National Science Foundation through grant IIS-1451172.

**References**


Robert St. Amant is an associate professor in the Computer Science Department at North Carolina State University. His research interests include human-computer interaction, intelligent user interfaces, and cognitive modeling. St. Amant has a PhD in computer science from the University of Massachusetts. Contact him at stamant@ncsu.edu.

David L. Roberts is an assistant professor of computer science at North Carolina State University, where he directs the Computational Intelligence and Interactive Games Research Lab. His research interests lie at the intersection of behavior, data, and computational modeling, with a particular emphasis on the role of computation in understanding and influencing behavior. Roberts has a PhD in interactive computing from the Georgia Institute of Technology. His work has been featured in national and international media outlets such as The Wall Street Journal, PBS Newshour, IFLS!, and the BBC. Contact him at robertsd@csc.ncsu.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
One in three Web users looks for medical information on social networks (http://bit.ly/wiredarabspring), and more than 50 percent of users surveyed consume news on social networks (http://bit.ly/pewsnsnews). Twitter and Facebook were prominent platforms used for disseminating information and organizing protests during Arab Springs, Occupy Wall Street, and similar events. Social data also plays a critical role in helping with coordination during natural disasters. Social networks have therefore not only changed the landscape for communicating and sharing information — they have also become a major source for users consuming information.

The popularity of social networks has led to an increase of user-generated content on the popular platforms. Facebook and Twitter together generate more than 5 billion microblogs per day. Because users consume information from these platforms, the overwhelming amount of content generated brings to mind Herbert Simon’s famous quote: “a wealth of information creates a poverty of attention.” In turn, the growth in the volume of content has often drawn criticism of information overload from consumers. As users of the Web, it’s important for us to realize that our dependence on information from these platforms will continue to grow, and hence, so should our focus on working towards making our lives easier in accessing the collected intelligence on these platforms, particularly by addressing the problem of information overload.

Researchers have addressed the challenge of information overload by developing information filtering systems that understand a group of users’ interests and deliver relevant content to them. Although these filtering techniques have been adopted for filtering spam in emails and delivering relevant news and articles to interested users, leveraging these techniques on social networks and building an efficient information-filtering system presents distinct challenges, due to social networks’ unique characteristics. Here, we consider how to address those challenges, using a crowd-sourced platform such as Wikipedia.

**Semantic Filtering for Social Data**

Consumers of social data face information overload. Although information filtering can help, challenges specific to the short-text and real-time nature of social networks remain. Harnessing knowledge bases from crowd-sourced platforms such as Wikipedia can help build an effective information-filtering system.
Semantic Filtering for Social Data

than traditional media outlets. As a result of this shift, journalists are now advised to monitor social networks for the latest news, and the disaster-management community uses social networks as a real-time communication platform, monitoring them closely to coordinate and manage situations during disasters.

To process the textual content of social networks and address information overload, any information-filtering technique used needs to handle two technical challenges: a lack of context and a dynamically changing vocabulary.

Lack of context. The task of analyzing textual content from social networks is fundamental to building an information-filtering system. This is necessary for understanding user interests from the posts they like or share on social networks, and for filtering posts that are relevant to their interests. For instance, consider the following post by a user on Twitter:

Example 1. “Great day for Chicago, Cubs beat Reds, Sox beat Mariners with Humber’s perfect game.”

The popular content-based assumption that users are interested in what they share, means we can infer that the author is interested in the Chicago Cubs, Cincinnati Reds, White Sox, and the Seattle Mariners. These topics we’ve identified then help us understand users’ interests, to filter future posts. This is notable because existing topic identification techniques perform significantly well on traditional and longer textual content such as blogs and news, but don’t perform well on short-text because they lack context for processing.

Dynamically changing vocabulary. Social networks can track topics such as a natural disaster, an election, or a sporting event. Changes in these topics are reflected by changes in the vocabulary used on social networks.

For example, the 2014 Indian election had various subevents associated with it that were emphasized during different times as the event unfolded. These include the announcement of prime ministerial candidates, issues regarding corruption in the political parties, and polls in different states. This topic was represented by multiple terms on Twitter, such as #modikisarkar, #NaMo, #VoteForRG, and #CongBJPQuitIndia, which evolved over time.

Social network conversations during natural disasters also exhibit significant changes over time. Disasters have been shown to go through phases as the situation evolves (for example: mitigation, preparedness, recovery, and response), and the conversation and language of social networks reflect this. During Hurricane Sandy, in particular, the representative hashtags evolved from #Frankenstorm and #Sandy at the start, to #StaySafe and #RedCross during the disaster, and #ThanksSandy and #RestoreTheShore after the hurricane.

Because more than 50 percent of users on social networks are interested in keeping up-to-date with the most recent topics and resort to social networks to do so, it’s important for filtering systems to adapt to changes happening in the real world. However, as social network platforms allow filtering based on keywords (or a combination of keywords), it becomes challenging for information-filtering systems to continuously monitor new, evolving keywords and filter relevant posts in real time.

Collective Semantics and LOD to the Rescue

We can address these challenges by developing techniques that leverage knowledge bases to enrich the semantics of short-text. Semantics is the relevant information inferred from knowledge bases related to the content in short-text to facilitate better understanding and processing. For instance, the semantic enrichment of the post in Example 1 involves the use of information associated with the Chicago Cubs. This information can be facts, such as Chicago Cubs are one of the Major League Baseball teams, or Jason Herward and Kris Bryant are its players. Such information can be found in structured knowledge bases on the Web.

Knowledge bases on the Web have grown in popularity due to the Linked Open Data (LOD) initiative and its focus on transforming the Web of hyperlinks into a Web of Data. Although there’s an abundance of knowledge bases on the Web, it’s important to select a relevant knowledge base (also referred to as an ontology or knowledge graph) to deal with the unique characteristics of social networking platforms. The knowledge base must satisfy two prominent requirements: first, broader coverage — because social networks handle a large set of diverse users, the filtering system requires that the knowledge base have a broad coverage of topics; and second, near real-time updates — social networks are real time in nature and mirror changes and activities in the real world. To utilize a knowledge base for filtering social data, it’s necessary for the knowledge base to be dynamically updated and also reflect the real world.

While knowledge bases on the LOD cloud, such as DBpedia, and Yago, encompass diverse topics and hence may satisfy the broad coverage requirements, these knowledge bases are updated infrequently — certainly not rapidly enough to cover a new, evolving event. Therefore, they’re unsuitable for real-time filtering of social data. However, one of the prominent sources of information for the aforementioned knowledge bases is Wikipedia, which is dynamically updated and reflects unbiased views of the real world in near real time. Wikipedia is an up-to-date collection of collaborative encyclopedic knowledge for most situations. In the remainder of this article, we discuss novel approaches that uses Wikipedia as a knowledge base by harnessing its semi-structure to address the challenges in filtering social data.
Enhancing Context Using Hierarchical Interest Graphs

Because the lack of context is what makes processing short-text challenging, Wikipedia can be exploited to improve context and identify topics. As humans, we naturally infer that the topics mentioned in Example 1 are related to Baseball, and hence the author might be interested in it. A similar inference can be made by machines using Wikipedia’s category structure, which is comprised of taxonomic knowledge that can be automatically extracted. Figure 1 portrays the relevant taxonomic knowledge from Wikipedia for topics mentioned in Example 1.

By using this hierarchical context from Wikipedia on social data, we can create an interest profile with a hierarchy—that is, a Hierarchical Interest Graph (HIG).

The HIGs encompass not only explicitly mentioned interests (such as the Chicago Cubs, Cincinnati Reds, and so on) but also others that are implicitly inferred from Wikipedia (such as Baseball and Major League Baseball). We adapt the spread activation algorithm to score each of the inferred interests, where the scores represent the extent of users’ interests.

HIGs add context derived from a knowledge base; hence, they address the issue of the lack of context. By using the HIG that encompasses implicit interests—for example, Baseball and its related topics—we can broaden the coverage of social data filtered for the user. In Table 1, based on the profiles generated for the author of Example 1, we can see the tweets that can be filtered using the knowledge-enhanced HIG for this user (Figure 1) versus interests that are directly extracted from the content. Example 1 mentions the Chicago Cubs, which is also explicitly mentioned in the first tweet in Table 1; hence, this tweet can be filtered by both profiles because it captures the Chicago Cubs as the author’s interest. However, the second and the third tweets in Table 1 are related to Baseball and can be filtered only using HIGs because they infer Baseball and related topics as the users’ interests from the Wikipedia category structure. To evaluate our approach for identifying hierarchical interests, we performed a user study involving 37 participants, which concluded that approximately eight of the top 10 interests were relevant to the user, and around 60 percent of these interests were implicit. To demonstrate the applicability of our approach, we applied the hierarchy of interests generated to recommend tweets for users. By augmenting content with knowledge-based user profiles (HIGs) we can improve the performance of tweet recommendation systems by more than 40 percent in comparison to other existing content-based recommendation approaches.

Harnessing Wikipedia’s Evolving Knowledge for Continuous Filtering

Social networks reflect the evolving topics of the real world by changes in the representative vocabulary used in posts. As we mentioned, during the 2014 Indian election the representative hashtags used on Twitter included #modikisarkar, #NaMo, #VoteForRG, and #CongBJPQuitIndia over time. To keep track of the evolution of the topic, the keyword filter must be up-to-date with representative vocabulary.

The approach we outline utilizes hashtags as keywords to filter information. Hashtags are a common way to represent topics and activities on social media. Hashtags that include more than one word are more informative vocabulary.

Social networks reflect the evolving topics of the real world by changes in the representative vocabulary used in posts. As we mentioned, during the 2014 Indian election the representative hashtags used on Twitter included #modikisarkar, #NaMo, #VoteForRG, and #CongBJPQuitIndia over time. To keep track of the evolution of the topic, the keyword filter must be up-to-date with representative vocabulary.

The approach we outline utilizes hashtags as keywords to filter information. Hashtags are a common way to represent topics and activities on social media. Hashtags that include more than one word are more informative.
Semantic Filtering for Social Data

Social networks such as Facebook and Instagram. We performed an analysis of hashtags extracted from 6 million tweets related to two dynamic topics — Occupy Wall Street and the Colorado shooting — and found that co-occurrence can be used as a starting point to continuously update hashtag filters. In other words, starting with a topic-relevant hashtag, such as #ows for Occupy Wall Street, we would be able to find other relevant hashtags (such as #owsla and #owsny). Figure 2 shows the co-occurrence graph of the hashtags relevant to the Colorado shooting and Occupy Wall Street.

Because detecting hashtags by co-occurrence alone can introduce noise, we take inspiration from the vision of continuous semantics⁹,¹⁰ and leverage Wikipedia as an evolving knowledge base to determine semantically relevant hashtags. The Wikipedia hyperlink structure is beneficial, because it evolves to reflect the changes in the dynamic topics. (For example, in Figure 3 we can see the change of links on Wikipedia for the 2014 Indian general election between 10 May 2010 and 20 May 2013.) The hashtags detected are used to continuously, periodically update the filter that collects relevant posts.¹⁰ A simulated real-time evaluation of this approach

Figure 2. Co-occurrence graph of hashtags relevant to (a) Occupy Wall Street and (b) the Colorado shooting. This is a hindsight analysis performed on manually curated data for Twitris.

Figure 3. Evolving Wikipedia hyperlink graph for the 2014 Indian general election as a topic, between 10 May 2010 and 20 May 2013.
for two dynamic topics from 2012 – the US presidential election and Hurricane Sandy – showed that the top five hashtags detected were able to improve coverage by retrieving new tweets with a high mean average precision of 0.92.

Using Wikipedia for information filtering has its own limitations. The updates for topics such as natural disasters and elections are quicker and the timeliness is comparable to social networks such as Twitter. However, for topics such as terrorist attacks and civil protests, the information propagation on social networks can be bottom up – that is, Twitter can reflect changes in the real world sooner than Wikipedia, and the lack of consensus on some topics (for example, whether democracy is suitable for a Middle Eastern country) can hamper the quality of timely knowledge that a Wikipedia-type knowledge source can provide. An analysis of Wikipedia updates and Twitter feeds shows that Wikipedia is comparably slower in receiving information.11

With the short-text on social networks, the network and the demographic attributes of users can enhance the performance of information-filtering systems by better understanding users. While the network dimension for filtering and recommending is well explored, inferring demographics of users on social networks has yet to be tackled, particularly due to the lack of demographic information shared on these platforms. In our recent work, we addressed the lack of location information12 for users on Twitter, which is one of the important attributes of demographics. The methodology identifies the city-level location of users, accessing Wikipedia as the source of background knowledge. The use of Wikipedia is just the tip of the iceberg, and the potential of knowledge bases – including the broader LOD and open data for the task of social data filtering – has yet to be well explored.

References

Amit Sheth is the LexisNexis Ohio Eminent Scholar and executive director of Kno.e.sis – the Ohio Center of Excellence at Wright State University. His research interests include smart data, physical Internet of Things, cyber and social Big Data; and semantic-cognitive-perceptual computing. Sheth has a PhD in computer and information sciences from Ohio State University. He’s an IEEE Fellow. Contact him at amit@knoesis.org or http://knoesis.org/amit.

Pavan Kapanipathi has recently completed his PhD at Kno.e.sis and will soon join as a research staff member at IBM T.J. Watson Research Center. His research interests include user modeling; the Semantic Web; recommender systems; and social data analysis. Kapanipathi has an MS in computer science from Wright State University. Contact him at pavan@knoesis.org.
From Zero to Hero: How Zero-Rating Became a Debate about Human Rights

Linnet Taylor • Tilburg Institute for Law, Technology, and Society

Zero-rating and the claim that everyone has a right to Internet connectivity is conflicting with the argument for net neutrality. Will free data services via mobile phones worldwide privilege a few content providers, or provide valuable access to information and services for low-income people?

Zero-rating has sneaked up on us. Until 2013 it was a bit of a marginal preoccupation, mainly an issue for those involved with the technopolitics of the Internet. People didn't fully understand net neutrality, or how zero-rating related to it. That was until Mark Zuckerberg announced that Internet connectivity was a human right, and that his new creation, internet.org, would provide free mobile data to those who couldn’t afford data plans. The debate really started when in India the project simultaneously met resistance from regulators and support from the country’s ruling party, causing India’s tech community to explode. Suddenly, instead of being an issue for specialists, zero-rating became what the Electronic Frontier Foundation called “the bleeding edge of net neutrality.”

The issue is in play worldwide, but some of the most vehement debate has been about its role in developing countries. Zero-rating has become a tumultuous international soap opera – as though the ill-starred lovers Zuckerberg and Narendra Modi, locked in passionate embrace, have been torn apart by India’s heartless regulators, various swashbuckling young activist organizations are locked in battle with corporate villains, and the international tech community has stepped in as the Greek chorus. All that’s missing is dragons. Meanwhile, low- and middle-income countries from Indonesia to Malawi are quietly getting on with consuming zero-rated services from a variety of different providers without too much fuss.

Zero-Rating and the Right to Connectivity

Besides forcing uncomfortable decisions on regulators, the zero-rating debate is also pushing rights-related buttons. Since the 1990s the information and communications technology (ICT)-for-Development field has focused on expanding technology access and connectivity into lower-income countries with the aim of fostering economic and human development. Much of this has involved outside intervention – in the shape of Universal Service Funds to facilitate access to the Internet for the poorest, development assistance to create and fund telecenter programs, and hundreds of corporate social responsibility (CSR) interventions by multinational technology companies to put computers in low-income communities. Then in the mid-2000s mobile phones came onto the scene and rapidly became seen as the answer to almost every development problem.

Against this background, the emergence of zero-rating created the perfect storm: the human right to connectivity became conflated with the technological and political debate about net neutrality. Suddenly the thorny technopolitical issue of what exactly constitutes a threat to net neutrality (the range of practices involved in the debate include slowing certain types of data or services down in comparison to others, offering some for free without slowing others down, offering cheaper deals on some services, or allowing inequalities in network traffic...
to reduce pressure on the system) became the much thornier issue of whether to deny disadvantaged people information about childhood illnesses or maternal health. So now those who complain that zero-rated services are creating a walled garden and stifling local innovation are against human rights and in favor of poverty and marginalization.

There are some problems with this argument. First, of course everyone should have connectivity, but connectivity per se doesn’t solve much. This is because the digital divide is structural and adapts to circumstances. It’s composed at least partly of language (English is the Web’s most widespread language, and those connecting in most less-common languages quickly run into difficulties when they seek more than basic information). The divide also relates to the user’s access to hardware and software, not just the Internet that runs on them. As Jonathan Zittrain pointed out back in 2008, some technologies provide for generative use of the Internet — exploring, creating content, engaging in debate, learning to code — while others just offer you a portal through which you can consume certain services while being surveilled by those providing them. Zero-rated services on mobile phones are more or less the epitome of non-generative technology.

**Won’t Zero-Rating Offer Vital Information and Services?**

Zero-rating, in the form in which it’s causing controversy — a package of services for free and the rest of the Internet for those who can pay — is also dangerously close to censorship, simply because those who really need free services are often unable to pay to get beyond them. The services that are being trumpeted as the real value of zero-rating — curated Web content related to education, job-seeking, and health — are also those that are most problematic as walled gardens. Wikipedia is inevitably a part of everyone’s education today, and is an excellent resource when used well, but what does a world look like where Wikipedia substitutes for any other educational Internet access? The kinds of services made available through zero-rating are great, and (mostly) come from reputable providers. However, they contribute to a particularly deterministic and reductive vision of mobile-for-development, which holds that just having mobile is fine, and that limited Internet access is also fine, because it’s the best that can be done under the circumstances.

The international development field has seen numerous versions of this argument over the last decades, one of the most notable being that people in the poorest places don’t need access to doctors and hospitals, they need nurses and basic clinic facilities. This is, however, a logical fallacy. Everyone needs nurses and clinics. They provide preventive healthcare, and deal efficiently with common small-scale problems. Imagine, however, that you’re in need of a cesarean delivery. Or suffering from Ebola. Or your child has appendicitis. In this position, you would be arguing pretty strongly for a structural approach based on the fundamental right to healthcare, and the redistribution of resources to get as close to it as possible.

For those who are unconvinced by this, consider that this argument has recently been made about roads. Matter, a startup based in the Silicon Valley (a place where people use roads almost every day) has said that drone delivery networks will allow the poorest countries to “leapfrog expensive road networks in much that same way that mobile phones have allowed them to leapfrog copper wires and landlines.” Think about that for a moment; perhaps again from the perspective of someone who needs a cesarean or appendicitis surgery — or just needs to get to school. The leapfrogging vision, like the efficiency-as-development vision, both work fine for specific cases when presented to gatherings of development economists. However, when you try to imagine your own world with no computer hardware, no Internet search functionality, no doctors, no hospitals, and no roads, it pretty soon becomes apparent that (to put it mildly) these solutions might create more problems than they solve.

While we’re momentarily seeing from a local perspective, it’s also worth thinking about the argument that zero-rating gets in the way of innovation. Experts at Research ICT Africa note that tilting the market could stifle competition from local social networks, app developers and content providers who cannot afford to pay providers to zero-rate their content or who do not have the market share or eyeballs on their products to make it attractive to operators to provide free data for their services.

When someone in a low-income area learns to code, gets hold of the requisite hardware, and actually sets up as a developer, they then have to find a market for their products. This market is likely to be local, and is therefore likely to compete with the kinds of offerings available from the paying “partners” involved in zero-rating. Not so much Unicef’s maternal-child health information, but certainly social networks and services, such as area-specific job search and other information resources.

Moreover, developers working in their own country are more likely to be able to address the Internet language barrier, which is currently a serious problem. Wikipedia might be available in 281 languages, but the African continent has at least 2,000. In Asia, Papua New Guinea alone has 820. Plus, a recent study shows that Wikipedia’s localness quotient (the extent to which people are writing or editing content about the place
from Zero to Hero: How Zero-Rating Became a Debate about Human Rights

where they live) is lower for poorer areas, and is also lower where contributors are writing in a language not spoken there. This suggests that zero-rated services that get in the way of people's ability to create and promote content that's relevant and comprehensible to those around them might not be promoting development so much as digital neocolonialism.

There are also social and political implications to this kind of market distortion. If zero-rating discourages the development of local social networks, information services, and other small-scale, area-specific content, it will also have an effect on the formation of collectives, on political activity, and on the type of economic opportunities available. Facebook is a fine platform for organizing, but should it be the only one? And what are the implications of concentrating all the communications of the poor and marginalized in a particular channel (or two, or three)? This kind of concentration creates fertile ground for alliances between online service providers and those interested in grabbing the votes of particular groups, influencing their behavior, or fomenting conflict.

All this suggests that the usefulness of zero-rating is dependent on context, on the ability of independent regulators to monitor who owns and licenses the providers, on how provision relates to political, social, and economic conditions, and on who has the ability to manipulate those conditions. In India we've seen an independent regulator shut down a zero-rating alliance between a massive global content provider and a nationalistic government, but not every regulator has the power or the appetite for conflict that India's has shown.

The Right to the Internet, versus the Right to Other Necessary Things

One argument frequently made in favor of zero-rating in developing countries is that it will introduce new users to the Internet. Of course it will, and it will also specifically increase uptake of data services in low-income regions. It's designed that way. It's worth asking, however, what we mean by "Internet users" and "uptake of data services" in this context. The kind of zero-rating being advocated for developing countries works on a basis whereby if people do a Google search using their zero-rated service and then click on a result, they'll get a message that they need a data plan to go further. Many of those who can afford a data plan will then buy one.

From the market's point of view, this is a victory for Internet adoption. However, many of those who can't afford a plan will also buy one, and this is less of a victory. Research ICT Africa's recent report shows that in South Africa low-income people often pay for mobile data at the expense of other things, including food and education. Advocates of a universal right to Internet connectivity rightly point out that people shouldn't have to choose between food and Internet access, and given that one of the arguments for zero-rating is that it can provide educational content, there's a particular irony in people spending money on educational apps instead of school.

This is an untenable conflict. But this is a problem particular to low-income regions, and must be addressed with an understanding of the structural factors involved. Mobile broadband costs 20 times as much in the world's poorest countries as it does in higher-income ones. Sub-Saharan Africa, where zero-rating is becoming a particularly popular response to a lack of Internet access, has the world's least affordable mobile data services at an average cost of more than 15 percent of per capita gross national income (GNI; see Figure 1). The only thing more expensive than mobile broadband in lower-income countries is computer-based broadband, which on average costs an eye-watering 30 percent of GNI per capita. The high-income-country equivalent would see the average American paying somewhere in the region of US$650 a month for a mobile data plan, plus a further $1,300 per month if they wanted to use the Internet at home. In Sub-Saharan Africa, despite the long-overdue landfall of new undersea cables, fixed broadband is actually becoming more expensive in many countries, leaving mobile connectivity to carry much of the burden. However, mobile isn't yet up to the task, because only 69 percent of the world has 3G coverage and the remaining 31 percent represents a lot of developing countries.

All this sounds like a strong argument for zero-rating. Free provision can just cut through the challenges of lower-income countries — monopoly providers, remote rural populations, tough geographic conditions for providing and maintaining infrastructure, governance in flux, variable regulation, and politicized access to markets for mobile operators — and offer people direct, free services using whatever connectivity is available. However, these difficult conditions are also a reason to look at prevailing assumptions about why functionality must be imposed from the outside. The most persuasive argument for zero-rating — that it gets an urgent job done quickly and efficiently — is actually an argument that has the potential to perpetuate poor conditions in telecom markets worldwide. Similar to the argument for vouchers as a way to remedy poor-quality public schools, zero-rating takes people out of the market and delivers them to preferred providers in ways that are neither sustainable nor democratic. It assumes that poor connectivity and unreasonably high prices are facts of life in poorer regions, rather than due to factors that can be influenced and
The Digital Citizen

Figure 1. Prepaid handset-based mobile-broadband prices (500 megabytes per month) as a percentage of gross national income (GNI) per capita in the Africa region, 2014. (Source: The International Telecommunication Union. The GNI per capita values are based on World Bank data.)

challenged by domestic and international pressure.

By arguing for a universal right to the Internet, we turn the Internet into something universal, decontextualized, and apolitical, whereas in fact it’s precisely the opposite. The Internet, whether delivered through mobile phones or cables, is composed of physical and logical infrastructure that must be created, negotiated, permitted, installed, regulated, and purchased in a long process of actions in order for a data plan to be offered by a provider. It’s both literally and figuratively embedded in national space. One function of zero-rating in developing countries is to take that national process of building and negotiating connectivity and turn it into The Internet. Once this disjunction has happened, it’s then usual to hear it argued that The Internet can Solve Poverty, Solve Education, or Solve Healthcare. The decontextualized, de-territorialized Internet can do any number of things, but — unfortunately — it can only do them in the abstract. For actual education to occur, rather than Education, Internet content must be translated, molded, and aligned to fit with what people need to know to progress in a particular place. It’s great to have massive open online courses (MOOCs), it’s great to have apps, and children are indeed miraculous, creative, curious creatures who will figure out how to make good use of any Internet access at all. However, as suggested before, try imagining your local school system has disappeared and your children are instead being offered Wikipedia in Swahili via a $10 smartphone. It’s interesting to watch them figure out how to use it, but it’s not quite what you had in mind when you paid your taxes. Oh, and if you want them to be able to click on a Google search result in their own language, you have to pay $650 a month.

The argument that any kind of connectivity will lead to increased capabilities for people is a difficult one to justify, despite the claims made by those planning to make billions from zero-rated services. It’s an argument better known as technological determinism, something that often prevails when technology and developing countries are mentioned in the same breath. It tends to lead to the logic that if you’re against [insert name of technological intervention here], you are against fundamental human rights and in favor of starving children. And while activists are defending themselves against this, the proponents of zero-rated services are making money. They make money from the massive expansion of their markets, from creating and then advertising to millions of new consumers, and from the long-term data trails those consumers create. Those data trails add up to massive and valuable datasets showing people’s previously uncharted preferences, behavior, and movements — not
to mention those of their social networks. When you see the longer-term picture, it looks much less as if the companies involved are doing people in developing countries a favor. Bruce Schneier has said that “surveillance is the business model of the internet.”

Zero-rating is more than free data: it’s a way to build long-term knowledge of markets, and to capture them as they mature. Despite claiming to provide the Internet as a tool for human development, it actually promotes it as a gateway drug.

So what will happen next? Probably what happens everywhere. People will gradually get online, whether using zero-rated or full services. They’ll watch football, watch Rihanna. They’ll scam, try to start businesses, and use Wikipedia for their school homework. None of this is new. However, those using the walled garden of zero-rating will gradually diverge a little from everyone else. They’ll live in more of a filter bubble, they’ll become more likely to replicate others’ use of the Web, they’ll be less likely to learn to manipulate the technology, and they’ll be less likely to use the Web in generative ways. These aren’t just problems with zero-rating; they’re also inherent in having mobile rather than computer-based connectivity, and in having slow or interrupted connectivity (for many Internet users in low-income countries, websites can take an entire session to load, and a .pdf can take all day). Gradually the paths of those potential Internet users will diverge and they’ll become low-income Internet users, capable of seeing less, doing less, and producing less. They might do fine as online microworkers, but they aren’t going to be the next Steve Jobs. He had the privilege of attending school – probably even using a road.

References


Linnet Taylor is an assistant professor of data ethics, law, and policy at the Tilburg Institute for Law, Technology, and Society (TILT). She researches the interface between Big Data, rights, and democratic representation worldwide. Taylor has a PhD in development studies from the Institute of Development Studies, UK. Contact her at l.e.m.taylor@uvt.nl; she also blogs at linnettaylor.wordpress.com.
I’ve previously written about emergent collectives (ECs) in the “Peering” column, and I mentioned the sharing economy (SE), otherwise known as the “gig economy.” Recently, I heard a good talk about the SE by Jan Marco Leimeister in his inaugural lecture at the University of St. Gallen. I was so struck by his evidence of this economic shift, which was part of my prediction in a previous article, that I read one of the supporting documents. Upon reading this analysis (really a survey and analysis of other studies) of the SE, I was struck by the relevance of the EC concept.

If you haven’t read, the features of an EC are:

- a network of information/function nodes that has minimal central control, and that’s largely controlled by a protocol specification,
- in which it’s easy for people to add nodes to the network,
- and where they have a social incentive to do so.

Examples of ECs are the WWW itself, Napster, Wikipedia, Facebook, Seti@home, and Linux. ECs are disruptive and difficult to predict because of the distributed nature of control. Previously I postulated, “Maybe we could say there are two types: social and economic emergent collectives.” I vaguely tied this to the idea of increasing self-employment: “We’re all becoming self-employed.” But reading makes me see that I missed a boat by not writing more about economic ECs.

Economic Emergent Collectives

In 2005, originally I focused on people sharing their resources with others for mutual benefit and to be part of something greater than themselves (www-cdr.stanford.edu/~petrie/revue). But apart from the examples of apps and Amazon’s Mechanical Turk, I totally failed to predict the rise of ECs, in which participants earned money facilitated by an intermediary platform. In fact, these examples are like ECs: we need only amend our third point of the definition to “and where they have a social or economic incentive to do so.”

Exemplars of SEs are Uber and Airbnb. But also, as I’ve pointed out, these are essentially ways of being self-employed, and include Bandwagon, Ridewith, Gett, Trippa, and Via in just the transportation arena. I mentioned webcamming and hotshoting, too (discussed further in the next section). If we look at what’s meant by “sharing” in all of these, it’s that an individual has a personal resource – either a service or tangible capital – that can be leveraged with a typically Internet-enabled intermediary platform. We might better call this “micro-capitalism” rather than “sharing.”

Immediately, you can see that not only crowdsourcing but also micro-loan systems are examples of ECs/SEs – because there’s a network that scales, as it’s largely automated by use of a protocol that allows people to easily add their nodes and gives them an incentive to do so. In fact, an excellent article on crowdsourcing distinguishes the kind of crowdsourcing in which people participate economically as “crowd work,” but all are some kind of EC.

Two Perspectives of the Same Phenomenon

So what I thought were different phenomena – ECs and increasing self-employment through SEs – are the same thing, seen from different perspectives. Yes, we’re increasingly self-employed, and the primary mechanism is SEs, because they’re ECs, which means they can scale.

Returning to the academic study of SEs, it’s easy to see that they miss the network aspect of SEs that the concept of ECs adds. Without this concept, the reason for the spread of these
disruptive technologies won’t be understood.

I also found that the studies covered by this meta-study were narrow in focus. For example, there were reports that one of the papers put the resources shared into four categories:

- spare time for performing errands or skilled tasks (such as Task Rabbit or Fiverr),
- spare time and cars to drive around customers (Uber or Lyft),
- extra rooms (Airbnb or Flipkey), and
- occasionally used tools and household items (Streetbank or Snap-Goods).

This is a narrow view of sharing and even presumes no overlap among resources shared. Hotshoting, for instance, is the “sharing” of trucks and trailers to haul “short loads” too small for a complete semi-trailer load. Webcamming is a way of selling sexuality virtually. All of these applications use some intermediary platform that facilitates someone adding a node to the network: a virtualized service, product, or, in general, a resource.

This EC aspect is in fact taken into account, to some extent, by the discussion of “multisided markets” that notes the ramifications of “P2P offerings are enabled by intermediary platforms that provide sufficient matchmaking services between resource providers and demanders (sic), often in exchange for a service or brokerage fee. The discussion here is good, but misses — as do the other good discussion points of crowdsourcing, trust and recommendation, and consumption-based pricing — the points of a network that’s easily added to and the scalable nature of the network control, which lets the network grow and thus be disruptive.

**Complex Tasks**

In general, I found Robin Knote and Ivo Blohm’s paper to be worthwhile reading, as it’s a first broad take on what’s an increasingly important phenomenon that will indeed allow more people to be self-employed, in possibly many areas at once. And one sentence really caught my attention: “Underlying calculation methods are especially interesting to investigate in case service offerings become more complex, as it is the case in (crowd) data driven calculation of multi-hop ride-sharing offers.”

This hints at the complex tasks I predicted but that haven’t occurred yet, which I’ve questioned. Virtual hitchhiking has been a standard way to travel in Germany for decades (https://de.wikipedia.org/wiki/Mitfahrzentrale), but it’s limited to point-to-point. Can such long-distance ride sharing catch on in other places, especially the US? There are of course such networks already, but it’s not yet clear that they’re spreading, perhaps because of cultural factors. For example, www.rdvouz.com does seem easy to use, and seems to fit the definition of an EC, but I haven’t heard of this particular technology disrupting long-distance bus services.

So let me ask you, the reader, whether you think multihop ride-sharing offers will be an evolution of current systems. Comment on this column at www.facebook.com/InternetComputingPeering — please.

**What’s Next?**

What can we expect next? One answer might be micro-manufacturing, because this also ties in with DIY (do it yourself). Because, just as you can create bitcoins at home with sufficient investment in compute power, you can, with sufficient capital, create a micro-factory for custom products with increasingly sophisticated 3D printers.

In fact, a rather interesting and controversial example of this is selling the plans, rather than the actual product, for guns, including semi-automatic rifles on the Web. One of the early-plan authors is already in a legal battle with the US government over what can and can’t be made legally with 3D printers.

That’s a fun example, but it’s not an EC. So what else is next? What clever network of 3D printers will enable more people to be self-employed, and maybe disrupt some part of commercial manufacturing?

One scenario for this is far-fetched right now. Suppose we create a network platform that allows various people to join together to make parts for say, custom bicycles. Shipping would be part of the outsourcing of all these tasks using some intermediary platform. And the more providers and customers that participate, the more powerful this virtual factory would become.

How far-fetched do you find this example? What are the barriers to such a system? What’s a better example of what’s next? Again, please do some of my work for me. Comment! Thank you in advance for sharing your insights.

_____

**References**


Charles Petrie teaches and coaches the topic of innovation in design thinking at the University of St. Gallen, Switzerland (http://dthsg.com/dt-at-hsg/). He retired as a senior research scientist from the Stanford University Computer Science Department. His research topics are concurrent engineering, enterprise management, and collective work. Petrie has a PhD in computer science from the University of Texas at Austin. He was a founding member of the technical staff at the MCC AI Lab, the founding editor in chief of IEEE Internet Computing, and the founding chair of the Semantic Web Services Challenge. He also manages the Black Rock City Municipal Airport 88NV. Contact him at petrie@cdr.stanford.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.
Early in my career, I was invited by colleagues at Stanford to join the faculty in electrical engineering and computer science. I had been a mathematics undergraduate at Stanford and took my PhD in computer science at UCLA. Having been on the Stanford campus for four years, I had some idea of the quality of the students there and concluded that I might not have much to teach them and turned down the offer. The faculty, including some who had written the textbooks I used in my computer science studies, started a telephone calling relay and by the end of the day, I had to “give in.” Of course, this turned out to be a good decision, because the initial design of the Internet was done with Robert Kahn while I was a Stanford assistant professor. Fast forward a few years, and the US Defense Advanced Research Projects Agency (DARPA) asked me to come to Washington to run the Internet and other packet network programs. I thought, “if I mess this up, everyone will know,” so I also turned down this offer. Another campaign convinced me to come and, of course, this proved to be a very successful six-year term.

But what’s the point of my telling you this, you might ask. Well, I wasn’t willing to take the risk of failing, but my colleagues convinced me otherwise. I counsel young people not to fear failure – it isn’t fatal, generally, and while you are young, you have plenty of time to recover. These anecdotes are about personal risk, but I want to explore a different kind of risk: collective risk.

This isn’t the first time I’ve written about this. It’s one thing to take personal risk, but it’s something else when someone puts others at risk. I am sure you can think of examples, such as drunk driving. What about “drunk programming”? Okay, that’s a bad joke. But in fact, when we write sloppy code, cut corners, ignore potentially exploitable bugs, and otherwise produce subpar software, we may well put many people at risk who are relying on the software to work reliably and safely.

As we encounter an Internet of Things, filled with software and capable of interacting locally and across the Internet, we’re sure to encounter a variety of risks — ranging from loss of privacy to serious malfunctions with physical consequences. An obvious question is how to respond to this kind of risk. What tools do we have to assess vulnerabilities and threats? How do we update software in devices that might have service lifetimes measured in years or even decades? How can we assure that control of these devices is confined to parties we authorize? How do we invoke and revoke authorizations (for example, for house guests)? How inconvenient is the mechanism for authenticating users? Do we have different mobile apps for each device? How do we configure scores or even hundreds of devices in a convenient and efficient fashion? Should law enforcement, fire, and medical personnel have access to residential and organizational devices, webcams, and sensors — and if so, under what conditions?

I’m sure that you can make up an even longer list of pertinent questions. Finding answers seems a lot harder. It seems to me inescapable that programmers eventually will have to be held accountable in some measure for the quality and safety of their work. Maybe not this year and maybe not in this decade, but at some point, it can’t be acceptable to put many people at risk without their knowledge or consent.
Assuming I’m not far into left field on this point, it’s important to ask how this responsibility (and liability) will manifest. There might be standards of software development practice that give evidence of responsible programming. I am somewhat skeptical of standardized software development methodologies, so I’m not necessarily advocating for them, but I do wonder whether practices might be developed or have already been developed that would move the professional towards improved trust and credibility. The increased focus on “cyber insurance” suggests that there’s growing awareness of the hazards of low software quality. Unfortunately, such insurance only deals with the potential financial risk that might accompany software failure(s). Insurance doesn’t, in and of itself, remove the bugs – and I hope that programmers, engineers, and their employer, aren’t confused on that point!

Vinton G. Cerf is vice president and chief Internet evangelist at Google, and past president of ACM. He’s widely known as one of the “fathers of the Internet.” He’s a fellow of IEEE and ACM. Contact him at vgcerf@gmail.com.
IEEE Computer Society 2016 Call for MAJOR AWARD NOMINATIONS
Help Recognize Computing’s Most Prestigious

IEEE Computer Society awards recognize outstanding achievements and highlight significant contributors in the teaching and R&D computing communities. All members of the profession are invited to nominate individuals who they consider most eligible to receive international recognition of an appropriate society award.

**Computer Entrepreneur Award**
*Sterling Silver Goblet*
Vision and leadership resulting in the growth of some segment of the computer industry.

**Technical Achievement Award**
*Certificate/$2,000*
Contributions to computer science or computer technology.

**Harry H. Goode Memorial Award**
*Bronze Medal/$2,000*
Information sciences, including seminal ideas, algorithms, computing directions, and concepts.

**Hans Karlsson Award**
*Plaque/$2,000*
Team leadership and achievement through collaboration in computing standards.

**Richard E. Merwin Distinguished Service Award**
*Bronze Medal/$5,000*
Outstanding volunteer service to the profession at large, including service to the IEEE Computer Society.

**Harlan D. Mills Award**
*Plaque/$3,000*
Contributions to the practice of software engineering through the application of sound theory.

**Computer Pioneer Award**
*Silver Medal*
Pioneering concepts and development of the computer field.

**W. Wallace McDowell Award**
*Certificate/$2,000*
Recent theoretical, design, educational, practical, or other tangible innovative contributions.

**Taylor L. Booth Award**
*Bronze Medal/$5,000*
Contributions to computer science and engineering education.

**Computer Science & Engineering Undergraduate Teaching Award**
*Plaque/$2,000*
Recognizes outstanding contributions to undergraduate education.

**IEEE-CS/Software Engineering Institute Watts S. Humphrey Software Process Achievement Award**
*(Joint award by CS/SEI)*
*Plaque/$1,500*
Software professionals or teams responsible for an improvement to their organization’s ability to create and evolve software-dependent systems.

**Deadline:** 15 October 2016
**Nomination Site:** awards.computer.org
**For more information visit:** www.computer.org/awards
Experience the Newest and Most Advanced Thinking in Big Data Analytics

03 November 2016 | Austin, TX

Big Data: Big Hype or Big Imperative?

BOTH.

Business departments know the promise of big data—and they want it! But newly minted data scientists can’t yet meet expectations, and technologies remain immature. Yes, big data is transforming the way we do—everything. But knowing that doesn’t help you decide what steps to take tomorrow to assure your company’s future.

That’s why May 24 is your real-world answer. Come meet the experts who are grappling with and solving the problems you face in mining the value of big data. You literally can’t afford to miss the all new Rock Stars of Big Data 2016.

www.computer.org/bda