IEEE Internet Computing

VOLUME 22, NUMBER 5 SEPTEMBER/OCTOBER 2018

Web Media

IEEE

IEEE computer society

www.computer.org/internet

IEEE computer society

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 275 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 30 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.

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928
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

revised 03 July 2018

IEEE Internet Computing

September/October 2018
Vol. 22, No. 5

www.computer.org/internet

TABLE OF CONTENTS

Feature Articles

Columns and Departments

ISSN: 1089-7801 Published by the IEEE Computer Society

Reflecting on Two Decades of Services Computing

Patrick C. K. Hung
University of Ontario Institute of Technology

M. Brian Blake
Drexel University

As young researchers more than 15 years ago, Patrick Hung and Brian Blake participated in the 2003 IEEE International Conference on Web Services. This year in July, Hung and Blake attended the most recent iteration of the conference, the 2018 IEEE World Congress on Services in San Francisco. Although the topics have varied over the decades, the 2018 Congress covered aspects of innovative services research and current and emerging applications in services computing. The Congress now contains seven different conferences in the areas of big data, cloud computing, edge computing, cognitive computing, the Internet of Things, web services, and services computing. Over lunch, Hung and Blake reflected on papers from this year's Congress and considered current trends in the community.

In 2003, *services computing* represented a relatively new research area that encompassed a new class of paradigms and technologies including web services, service-oriented architecture (SOA), business process integration and management, utility/grid computing, and autonomic computing. At the time, the IEEE Computer Society had officially launched the Technical Steering Committee for Services Computing (TCSVC). The discipline of services computing covers the science and technology of bridging the gap between business services and information technology services. This year's IEEE World Congress on Services had interesting contributions in four research categories: big data analytics/cognitive computing, mobile edge computing, machine learning, and robotic computing. Here, we discuss some of the papers from these areas and their contributions.

BIG DATA ANALYTICS

Big data is a term that has been gaining considerable attention in recent years. It describes a large amount of organized or unorganized data that is analyzed to make informed decisions or

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

evaluations. The data can be taken from a variety of sources including browsing history, geolocation, social media, purchase history and medical records. There are three main characteristics associated with big data:

1. *volume* is used to describe the vast amounts of data that is utilized by big data;
2. *variety* is used to describe the many different types of data sources used as part of a big data analytics system; and
3. *velocity* is used to describe the speed at which data is generated.[1]

Here, we share a few readings from the Congress on the latest findings.

Referring to data management and quality evaluation, Ikbal et al.[2] presented an across-the-board quality management framework. It includes a roadmap for data scientists that considers the assessment of quality as early as possible and end-to-end integration across the following areas:

- implementation of continuous quality improvement and enforcement mechanisms in quality management;
- specification of data quality metrics that should cope with the data's dynamic nature and its unconventional characteristics;
- development of new quality dimensions with specific measurement attributes for unstructured and schema-less data;
- enforcement of quality requirements, generation of quality reports and feedback to support assessment activities;
- development of automated real-time dashboards for data quality monitoring;
- application of higher degrees of statistical proof in different data quality evaluation processes including sampling, regression, correlation, and matching;
- development of effective quality outcome predictions; and
- evaluation of the quality of representative sets of data samples and generation of quality models to apply to the whole data.

It's widely accepted that quality is the most important foundation to support big data analytics.

For big data analytics, researchers establish approaches to mine and discover unknown patterns and insights from huge volumes of raw data.[3] Big data analytics has become very popular in the areas of marketing and customer-relationship management. Many industries have adopted the use of big data analytics and are experiencing fantastic results. For example, the healthcare, retail, insurance, and telecommunications industries have all displayed the endless possibilities of implementing big data into their operations.[1]

Khalajzadeh et al.[3] studied data analytics software tools for domain experts who are not computing specialists. The tools have the following functions:

- to cover data preprocessing operations such as cleaning, wrangling, anomaly detection, and so on;
- to incorporate a variety of algorithms for each stage of data processing, modeling, and evaluation processes; and
- to cover software development life cycle (SDLC) stages, including business problem descriptions, requirements, design, implementation, testing, and deployment.

A research topic related to big data analytics is sentiment analysis. Sentiment analysis techniques determine the overall sentiment orientation for topics discussed in the text as positive, negative, or neutral, while emotion detection from text identifies the categories of emotions the text expresses.[4] Analyses of text using emotional categories have proven valuable with the growth of social media tools, such as Twitter, for communication and collaboration. Traditional sentiment analysis only visualizes an aggregation of opinions expressed in the content, while neglecting the presence of the creators of the content and the impact of their varying levels of participation. Hemmings-Jarrett et al.[5] addressed this gap and concluded that differences in the level of user participation potentially impact the samples extracted for sentiment analysis and interpretation in their study.

MOBILE EDGE COMPUTING

Mobile edge computing is a network architecture concept that enables the cloud computing service environment at the edge of the cellular network by running applications and performing related processing tasks closer to the cellular customer. Mobile edge computing is designed to decrease latency and network congestion for mobile users. Zhang et al.[6] presented a quality of experience (QoE) aware control plane for adaptive streaming service over mobile edge computing infrastructures with the following features:

- a timeslot system with a look-ahead window for calculating the cost of edge node switch and video quality adaption (to balance network load and reduce latency);
- conducting service adaption via a set of cooperative action components running on client devices, edge nodes, and center nodes (to ensure a smooth viewing experience); and
- constructing a flexible QoE model and extending the scope and meaning of user-perceived experience.

Edge servers are usually deployed at the edge of the network so that computation is performed at the proximity of data source. This has two advantages: on downstream data, edge servers play a role of cloud service provider, making computing resources close to end users so that the latency of service request can be very low; and on upstream data, it helps to improve the network transmission on the core network. Li and Wang[7] studied the problem of energy-aware edge server placement as a multi-objective optimization problem and found a more effective placement scheme with low energy consumption.

MACHINE LEARNING

Intelligence in computing is essential to achieve service excellence for the ever more complicated requirements of the rapidly evolving global environment, as well as to discover useful patterns among the vast amount of data. This involves knowledge from various disciplines such as computer science, industrial and systems engineering, management sciences, operation research, marketing, contracts, and negotiations. It also involves cultural transformation and integration methods based on beliefs, assumptions, principles, and values among organizations and humans. For example, machine learning has been used in recent years for processing and analyzing service-oriented architecture, providing insights to businesses and policymakers for making intelligent decisions. More recently, deep learning technology promises to further revolutionize such processing, leading to better and more accurate results. Deep learning employs software tools from advanced analytics disciplines such as data mining, predictive analytics, text mining, and machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or nonlinear transformations. As such, the processing and analysis of deep learning applications present methodological and technological challenges together with opportunities. For example, Ishtiaq et al.[8] presented a semi-supervised clustering-based diagnosis recommendation model in healthcare via machine learning techniques based on an unstructured textual dataset.

Referring to intelligent transport systems, Abbas et al.[9] presented a short-term road traffic density prediction based on long short-term memory (LSTM) neural networks. The model is trained by using traffic data collected by the Motorway Control System in Stockholm, that monitors highways and collects flow and speed data per lane every minute from radar sensors based on partitioning the road network into stretches and junctions with one or more LSTM neural networks. On the other side, Duan et al.[10] discussed a neural network-based method to simulate the cognitive process of how human beings read Earth science articles and identify implicitly cited dataset entities from the articles.

Deep learning employs software tools from advanced analytics disciplines such as data mining, predictive analytics, text mining, and machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or nonlinear transformations. However, the processing and analysis of deep learning applications present methodological and technological challenges. Further, deep learning applications

are advantaged by a rise in sensing technologies as witnessed by both the number and rich diversity of sensors ranging from smartphones, personal computers, and health tracking appliances to the Internet of Things (IoT) technologies. These technologies are designed to give contextual, semantic data to entities in a ubiquitous environment that could apply intelligence to decision making. Recently, deep learning technologies have been applied to service-oriented computing. For example, Cai et al.[11] presented an example of applying advanced deep learning techniques on a large-scale, geo-tagged, and image-based dataset measuring urban tree cover using Google Street View (GSV) images to efficiently estimate important urban metrics, particularly in deep convolutional neural networks.

Referring to the brain-computer interfaces, Bellman et al.[12] described an experiment to determine if modern machine learning techniques could be used to accurately detect and classify unaware and aware facial recognition. The experiment consisted of participants viewing a variety of images. Over a period of three phases across two days, participants were first trained on a number of images that they were to implicitly learn for unaware recognitions on the following day. On the second day of the experiment, participants were shown these implicitly learned images, among others including a single memorized face for aware recognition, and then the electroencephalogram signals were recorded for later analysis.[12]

ROBOTIC COMPUTING

Robotic computing is a branch of artificial intelligence (AI) technologies and their synergistic interactions that enable and are enabled by robots. James Kuffner at Google coined the term "cloud robotics" to describe a new approach to robotics that takes advantage of the Internet as a resource for massively parallel computation and real-time sharing of vast data resources. For example, Li et al.[13] investigated the task assignment and scheduling in collaborative cloud robotic systems (CCRS), in which robotic agents can work cooperatively, not only by sharing their processing resources with each other but also by supporting cloud services, making them more intelligent, efficient, and knowledgeable.

CCRS is a technical solution to fulfill complex tasks, such as multi-robot Simultaneous Localization And Mapping (SLAM). However, the challenges stem from not only the computation complexity of large-scale map merging but also the inefficiency of enabling parallel computing in this process, which is indispensable to make available the frontier of computing technology, such as cloud infrastructure. For example, Zheng et al.[14] presented a scalable real-time multi-robot visual SLAM framework based on the cloud robotic paradigm that can distribute the SLAM process to multiple computing hosts in a cluster, which enables map building in parallel. Further, Silva Filho et al.[15] described a robotic platform developed within Baker Hughes, a GE company (BHGE) and GE Global Research Centers (GE-GRC), discussing its use in an industrial inspection case study for remote methane inspection in oilfields.

CONCLUSION

Services computing continues to evolve. We were delighted that the area continues to thrive both in research and in industry. Services computing is integrated into the physical world more today than ever before. The future of this area will continue to be connected to big data in addition to practical applications in artificial intelligence and human-centered computing.

We certainly hope to be having this conversation again in another decade but can only imagine how the area will evolve. This month's issue has been crafted from a group of papers that overlap human-centered interaction with the Internet and Web Media. We hope you enjoy the papers selected for this issue.

REFERENCES

1. *Big Data Applications and Use Cases*, Patrick C.K. Hung, ed., Springer International Series on Applications and Trends in Computer Science, Springer, 2016.
2. I. Taleb, M.A. Serhani, and R. Dssouli, "Big Data Quality: A Survey," *Proceedings of the 2018 IEEE International Congress on Big Data*, 2018, pp. 166–173.
3. H. Khalajzadeh et al., "A Survey of Current End-user Data Analytics Tool Support," *Proceedings of the 2018 IEEE International Congress on Big Data*, 2018, pp. 41–48.
4. A. Agarwal et al., "Sentiment analysis of Twitter data," *Proceedings of the 2011 Workshop on Languages in Social Media* (LSM 11), 2011, pp. 30–38.
5. K. Hemmings-Jarrett, J. Jarrett, and M.B. Blake, "Sentiment Analysis of Twitter Samples that Differentiates Impact of User Participation Levels," *Proceedings of the 2018 IEEE International Conference on Cognitive Computing*, 2018, pp. 65–72.
6. L. Zhang, S. Wang, and R.N. Chang, "QCSS: A QoE-aware Control Plane for Adaptive Streaming Service over Mobile Edge Computing Infrastructures," *Proceedings of the 2018 IEEE International Conference on Web Services*, 2018, pp. 139–146.
7. Y. Li and S. Wang, "An energy-aware Edge Server Placement Algorithm in Mobile Edge Computing," *Proceedings of the 2018 IEEE International Conference on Edge Computing*, 2018, pp. 66–73.
8. I. Ahmed et al., "Diagnosis Recommendation Using Machine Learning Scientific Workflows," *Proceedings of the 2018 IEEE International Congress on Big Data*, 2018, pp. 82–90.
9. Z. Abbas et al., "Short-Term Traffic Prediction Using Long Short-Term Memory Neural Networks," *Proceedings of the 2018 IEEE International Congress on Big Data*, 2018, pp. 57–65.
10. X. Duan et al., "A Neural Network-Powered Cognitive Method of Identifying Semantic Entities in Earth Science Papers," *Proceedings of the 2018 IEEE International Conference on Cognitive Computing*, 2018, pp. 9–16.
11. B.Y. Cai et al., "Treepedia 2.0: Applying Deep Learning for Large-scale Quantification of Urban Tree Cover," *Proceedings of the 2018 IEEE International Congress on Big Data*, 2018, pp. 49–56.
12. C. Bellman, M.V. Martin, and S. MacDonald, "On the Potential of Data Extraction by Detecting Unaware Facial Recognition with Brain-Computer Interfaces," *Proceedings of 2018 IEEE International Conference on Cognitive Computing*, 2018, pp. 99–105.
13. S. Li et al., "Latency-Aware Task Assignment and Scheduling in Collaborative Cloud Robotic Systems," *Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing*, 2018, pp. 66–72.
14. P. Zhang et al., "Cloud-based Framework for Scalable and Real-time Multi-robot SLAM," *Proceedings of the 2018 IEEE International Conference on Web Services*, 2018, pp. 147–154.
15. R.S. Filho et al., "Semi-Autonomous Industrial Robotic Inspection: Remote Methane Detection in Oilfield," *Proceedings of the 2018 IEEE International Conference on Edge Computing*, 2018, pp. 17–24.

ABOUT THE AUTHORS

Patrick C. K. Hung is a professor at the University of Ontario Institute of Technology. His research interests include services computing, toy computing, robotic computing, security and privacy. Hung received a PhD in computer science from the Hong Kong University of Science and Technology. Contact him at patrick.hung@uoit.ca.

M. Brian Blake is the provost of Drexel University and the editor in chief of *IEEE Internet Computing*. His research interests include service-oriented computing, workflow, and collaboration. Blake received a PhD in software engineering from George Mason University. Contact him at mb3545@drexel.edu.

Serverless Is More: From PaaS to Present Cloud Computing

Erwin van Eyk
Delft University of Technology

Lucian Toader
Vrije Universiteit Amsterdam

Sacheendra Talluri
Delft University of Technology

Laurens Versluis
Vrije Universiteit Amsterdam

Alexandru Uță
Vrije Universiteit Amsterdam

Alexandru Iosup
Delft University of Technology and Vrije Universiteit Amsterdam

Editor:
George Pallis
gpallis@cs.ucy.ac.cy

In the late 1950s, leasing time on an IBM 704 cost hundreds of dollars per minute. Today, cloud computing—using IT as a service, on demand with pay-per-use—is a widely used computing paradigm that offers large economies of scale. Born from a need to make platform as a service (PaaS) more accessible, fine-grained, and affordable, serverless computing has garnered interest from both industry and academia. This article aims to give an understanding of these early days of serverless computing: what it is, where it comes from, what the current status of serverless technology is, and what its main obstacles and opportunities are.

The 1950s saw the emergence of two technologies that are currently shaping the world: containerization in shipping and time sharing in computing. By allowing shipping to become standardized and automated, containerization gave rise to manufacturing and retail ecosystems, and ultimately to the economic phenomenon of globalization.[1] By enabling multiple clients to share the same physical infrastructure, time sharing gave rise to cloud computing and the modern digital ecosystems, which are key drivers for growth in knowledge-based societies.[2]

Whereas few companies or people could afford the cost of time-sharing services and paid dearly for simple computer simulations in the late 1950s, today more than 80% of companies, along with many private individuals, use the hundreds of services accessible through cloud computing.[3,4] Following with remarkable regularity the evolution observed in the history of containerization, cloud services have adapted to offer better-fitting containers that require less time to load (boot) and to provide increased automation in handling (orchestrating) containers on behalf of the client.

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

Serverless computing promises more: to achieve full automation in managing fine-grained containers. Already, IT spending on serverless computing is expected to exceed $8 billion per year, by 2021.[5] To understand what serverless is and what it can deliver, we trace the evolution of computing technology that has given rise to serverless computing, analyze the current status of serverless technology, and identify the main obstacles and opportunities we see in delivering on its promise.

What is serverless computing? We have proposed the following definition:

Serverless computing is a form of cloud computing that allows users to run event-driven and granularly billed applications, without having to address the operational logic.[6]

This definition places serverless as a computing abstraction, partially overlapping with platform as a service (PaaS). With serverless, developers focus on high-level abstractions (e.g., functions, queries, and events) and build applications that infrastructure operators map to concrete resources and supporting services. This effectively separates concerns, with developers focusing on the business logic and on ways to interconnect elements of business logic into complex workflows. Meanwhile, service providers ensure that the serverless applications are orchestrated—that is, containerized, deployed, provisioned, and available on demand—while billing the user for only the resources used. These separated roles have also emerged for physical containers, with manufacturers and retailers in the role of developers, and shipping lines in the role of service providers.

Clients of serverless computing could use the function as a service (FaaS) model, which we define this way:

Function as a service (FaaS) is a form of serverless computing in which the cloud provider manages the resources, lifecycle, and event-driven execution of user-provided functions.[6]

With FaaS, users provide small, stateless functions to the cloud provider, which manages all the operational aspects to run these functions. For example, consider the ExCamera application, which uses cloud functions and workflows to edit, transform, and encode videos with low latency and cost.[7] A majority of the tasks in these operations can be executed concurrently, allowing the application to improve its performance through parallelizing these tasks.

To deploy ExCamera using the traditional infrastructure as a service (IaaS) model, a user would need to spin up virtual machines (VMs), provision them, orchestrate the workflows, manage resources as needed, and manage the variety of dynamic issues (e.g., faults and inconsistencies). This would require considerable expertise and continuous effort in orchestrating ExCamera, and yet result in significant amounts of underutilized but paid-for resources. Instead, by leveraging serverless computing, ExCamera defers the operational complexity to the cloud provider, using FaaS to manage the operational lifecycle of the individual video tasks.

However promising, serverless computing is still an emerging technology. Understanding how applications such as ExCamera can leverage it requires finding answers to questions such as these:

- What are the computer technologies underlying serverless?
- What is the status of the current serverless technology?
- What can we expect from the field of the serverless computing in the foreseeable future?

We address these questions with a threefold contribution. First, we identify the concepts leading to serverless computing, with deep historical roots in concrete and abstract innovations in computer science. Second, we analyze the current state of the technology and the complex technological ecosystems it consists of. Finally, we identify and analyze important obstacles and

opportunities in this emerging field that we—as a community—need to address to make the serverless promise a reality. We conclude with a forewarning question: can we reproduce the successes and avoid the downsides of physical containerization?

THE LONG ROAD TO SERVERLESS

In this section, we analyze the evolution of computer technology that led to serverless computing—going back to the 1960s. All the breakthroughs indicate that serverless computing would not have been possible a decade ago, when it would have missed enabling technologies such as the distinction between IaaS and PaaS (standardized by NIST), fine-grained containerization (e.g., Docker), and even a diverse set of applications.[3] In Figure 1, we distinguish six main dimensions of these critical breakthroughs that together led to the emergence of serverless.

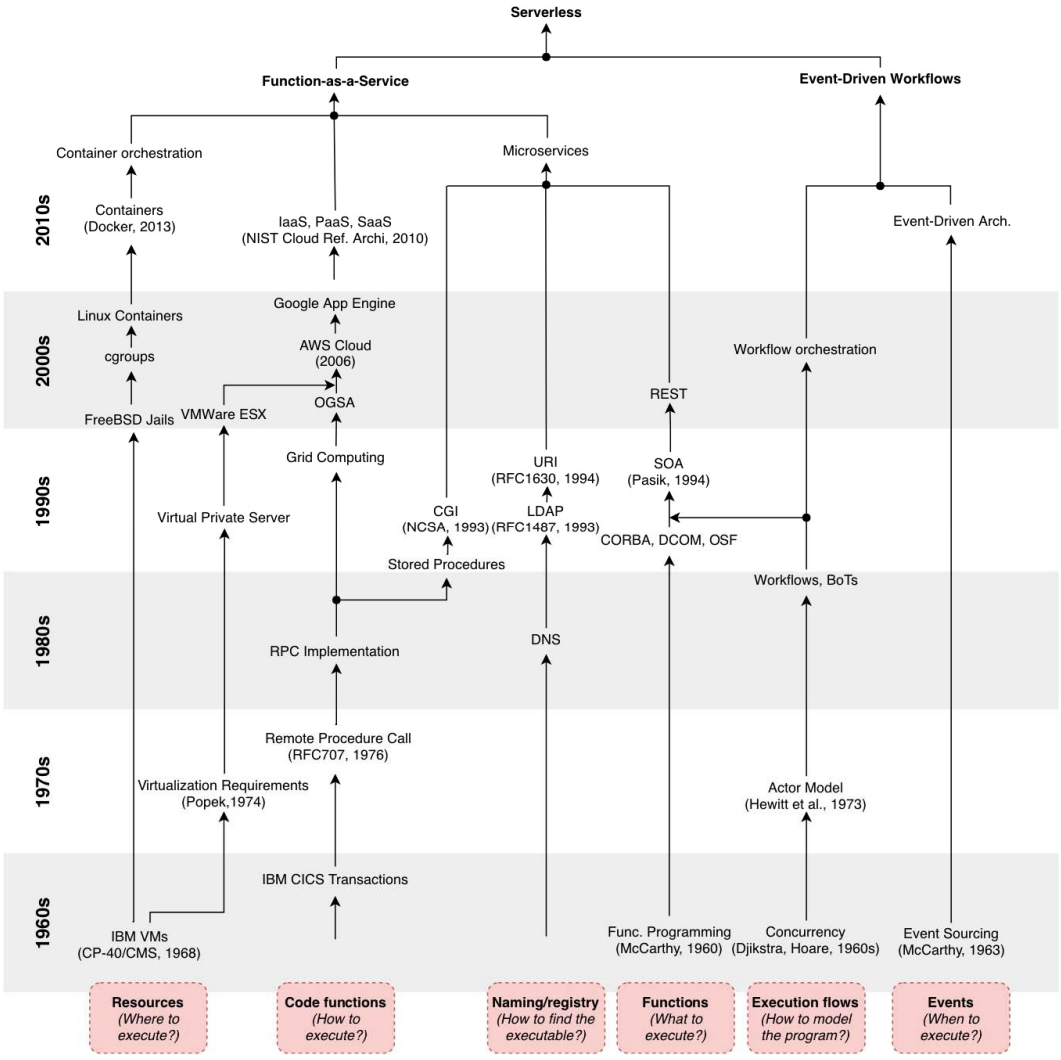

Figure 1. A history of computer science concepts leading to serverless computing.

CONTAINERIZED RESOURCES

Complementary to time sharing, virtualization abstracts away the physical machine to reduce the operational effort and to allow the same physical resources to be shared across multiple applications and users (multiplexing). Although associated in recent memory with VMware's ESX technology (2001), virtualization was invented much earlier; it was used in production in late-1960s

IBM mainframes. Virtualization was finally conceptualized nearly a decade later.[8] With the emergence of the Internet in the 1990s, these early concepts of virtualization were introduced online to enable shared hosting through virtual private servers. Soon after the release of ESX, cloud computing emerged, making virtual resources available over the Internet.

Like their physical counterparts, digital containers protect their content from external abuse. They do this by adding a layer of abstraction over the resources provided by the system. FreeBSD added jails—independent partitions of the system with their own directory subtree, hostname, IP address, and set of users. Linux followed with cgroups (2006), a mechanism to group processes and to assign each group separate resources. The Linux Containers project (LXC, 2008) bundled cgroups and kernel namespaces, along with better tooling. Built on LXC, Docker (2013) offered convenient container orchestration, fostering an entire ecosystem based on digital containers.

Serverless computing is the latest result of this long-term process of defining virtualization abstractions, to eliminate concerns related to server provisioning and management. Although it exposes abstract resources (e.g., functions) to the user, these are mapped to concrete resources (e.g., containers), continuing the transition from "bare metal" to "bare code."

Code as Functions

The ability to execute arbitrary "cloud functions" is essential to serverless computing.

Technology has emerged and reemerged often for running domain- and context-specific remote functions. We can trace this concept to as early as 1968, when, with IBM's Customer Information Control System (CICS), users were able to associate user-provided programs with transactions. RPC (specification in 1976, implementation in 1984) enabled the invocation of arbitrary procedures located in remote systems, over a communication network. Derived from RPC, stored procedures for databases (1980s) and Common Gateway Interface (CGI) scripts for webservers (1990s) aimed to bring support for executing functions to specific domains. Google App Engine—with other PaaS platforms following its example—started allowing users to asynchronously execute arbitrary tasks in the background.

In contrast to these context-specific implementations, serverless computing aims to provide a full abstraction for arbitrary, event-driven execution of generic functions.

Naming and Discovery

Managing and invoking services, including functions, depends on being able to name and discover services. Derived from a long line of technological innovations, current approaches follow the pathbreaking concepts of Lightweight Directory Access Protocol (LDAP, 1993) and URI (1994). LDAP uses naming and properties and enables distributed directory services over TCP/IP. URI provides unique identifiers for resources, encoded as character strings.

Serverless extends naming and discovery with function versioning and aliases—e.g., offered by Amazon Web Services (AWS). With versioning, it is possible to work with different immutable versions of a function simultaneously. Aliases are mutable pointers to a version and can be used to transition a version from one stage to another (e.g., from development to production) without changing the deployed application.

Functions as Computation

Serverless computing relies on the concept of function as computation, which stems from a long tradition of ever-higher-level abstractions and specialization in computer science.

Functional programming[9] departed from procedural programs, to allow the developer to manage abstract data types and control flows, instead of the concrete details of memory and processors. The application of object-oriented principles to distributed systems lead to the creation of

DCOM (Distributed Component Object Model), CORBA (common object request broker architecture), and OSF (the Open Software Foundation) in the 1990s. In the 2000s, we climbed up the specialization ladder by contextualizing and interconnecting services—e.g., through service-oriented architecture[10] (SOA) and architectures based on REST (Representational State Transfer).

These previous developments gradually led to microservices: self-contained applications providing specific functions over well-defined protocols.[11] Continuing this trend, serverless development is effectively a hyperspecialization of services.

Execution Flows

Serverless computing depends on the ability to coordinate execution flows.

Concurrency[12] has been an early and vital model for the evolution of computing, allowing multiple processes to make progress at the same time, while remaining under the developer's control. This model has many applications, and many other models are rooted in concurrency, including generalized processes, threads, and actors.[13]

Over the past two decades, we have moved toward a declarative form of expressing concurrency. Workflows declare the structure of applications, leaving the concrete execution and synchronization of workflow tasks to the runtime system. This model has a multitude of applications[14] and underlies our view of serverless computing.

Events to Trigger Functions

The first computer programs were synchronous, carefully crafted to follow a particular code path. This model made programs difficult to create and modify and less robust to changing conditions. Soon, event sourcing addressed the need to record, order, and respond to requests for state changes.

With the proliferation of high-level languages and advanced operating systems, the concept of linking disparate computation together with special communication constructs took hold. Device drivers were early examples of this event-driven programming. With the rise of the Internet, event-driven distributed systems became widely used—with events mapped intuitively to the asynchrony of real-world networks.

Indeed, in modern systems, event-based protocols allow systems in an ecosystem to communicate without excessive dependence on the implementation details of each individual system. Owing to its highly networked nature, serverless computing is apt to leverage this idea, through well-defined event protocols and ways to manage events—for example, by using message queues.

SERVERLESS NOW

In this section, we discuss how ExCamera could use serverless computing. We explain the current and emerging technological ecosystem for serverless, and detail the expected benefits from using serverless: better resource management, scaling, and more insight and control.

The State of Serverless Technology

To execute parts of the workloads using small serverless functions, ExCamera parallelizes video transcoding using AWS Lambda, one of the many FaaS platforms.[15] These various platforms differ in focus, target domain, assumed model, and architectural decisions. Next to the closed-source FaaS platforms, addressing the lack of insight and vendor lock-in, several open-source platforms have emerged, including Apache OpenWhisk, Fission, and OpenLambda.[16]

To address the complexity of working with many different FaaS platforms and their incompatible APIs, the community has focused on informal standardization. Serverless frameworks—e.g., Apex or the Serverless Framework—provide a common programming model that enables easier, platform-agnostic development, along with better interoperability of functions.

Much serverless tooling already exists to allow developers to defer nonessential tasks, sparking the emergence of a diverse ecosystem of (serverless) services, from serverless databases to monitoring to security. For example, workflow engines, such as Azure Logic Apps, Fission Workflows, and PyWren,[17] abstract away the complexity of networking in the composition of higher-order functions and services.

Benefits of Current Serverless Technology

Serverless computing promises more value than other cloud operations: equal or better performance while reducing the operational costs of applications. This has led industry—rather than academia—to drive the initial development and adoption of this paradigm. Figure 2 illustrates the main case for serverless computing.

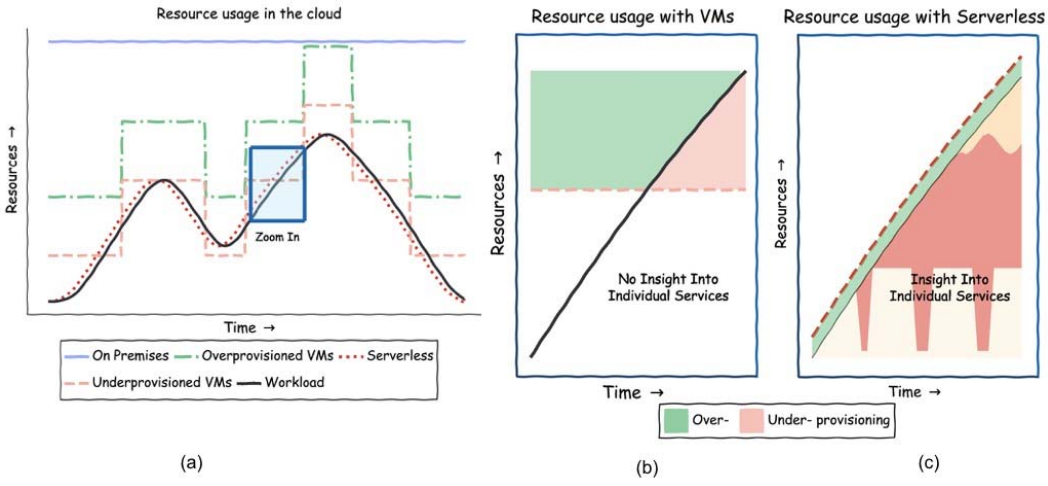

Figure 2. A case for serverless computing: higher resource utilization, finer granularity, and more detailed control than with container-based or self-hosted computing. (a) Resource utilization in the cloud. (b) For the zoom-in, resource utilization with virtual machines (VMs). (c) For the zoom-in, resource utilization with serverless computing.

Benefit 1: Improved Resource Management

In the traditional cloud model, the user is responsible for selecting and deploying the concrete resources. To avoid overburdening the user with options, the range of options is generally limited to large, multifunctional resource types (e.g., VMs or containers). Applications rarely fit these resources, and, to mitigate the overhead incurred by the large, general-purpose resources, applications are coarse-grained.

As illustrated in Figure 2a, coarse-grained applications lead to inaccurate autoscaling decisions, causing severe under- or over-provisioning. In contrast, serverless computing means applications are fine-grained, which means the cloud provider can more closely match abstract resource demand to actual system resources.

Benefit 2: More Insight and Control

In the traditional model, the user is responsible for deploying, monitoring, and other operational tasks related to the lifecycle of coarse-grained applications (see Figure 2b). However, many cloud users do not have the necessary expertise. Moreover, the operators lack context, so they have to make autoscaling decisions without accurate profiling or insights from the deployed applications.

With serverless, the increased responsibility for the operator gives more insight and control. Operators select the resources; deploy and provision resources; implement and control the monitoring of resource usage, workload intensity, and application behavior; and can autoscale or migrate the application. They can profile and model the granular services constituting the serverless application (see Figure 2c), offer this information to users, and improve the decisions made with these insights.

Benefit 3: Granular Scaling

In the traditional model, applications consist of large, multifunctional VMs with multiminute provisioning times. These VMs act as black boxes and thus are difficult to model and predict for operators. Although applications are typically bottlenecked by only one of the resources in one part of the application, the operators can only scale the entire application to resolve the bottleneck. Eliminating some of these issues is possible but requires the user to rearchitect the application, typically as microservices. The effectiveness depends highly on the user's expertise[11]—most users cannot benefit.

With serverless, the operator can better scale the individual, granular services or functions, using deep insights. The contrast between Figure 2b and Figure 2c illustrates this situation.

Other Benefits

Other benefits motivate the adoption of serverless computing. The shift from capital expenses to operational expenses more accurately aligns the costs to the actual business processes. The independent services allow teams to choose the right tools and dependencies for a use case without impacting other parts of the system and organization. The high-level abstraction allows software developers to iterate on these distributed systems more quickly, while limiting the need for extensive expertise in distributed systems. Etc.

PERSPECTIVES ON SERVERLESS

Although serverless computing already offers many benefits, many obstacles could inhibit its further adoption. In joint work with the Standard Performance Evaluation Corporation RG Cloud Group (https://research.spec.org/working-groups/rg-cloud.html), we have identified more than 20 detailed challenges and opportunities for serverless computing.[5,18] Here, we identify the top five obstacles and opportunities arising from them (see Table 1).

Table 1. Obstacles and opportunities for serverless computing.

Obstacle	Opportunity
Fine granularity and cost	Nontrivial resource management, workflows of functions, orchestration, fine-grained "pay-per-use" pricing, optimizing cost–performance tradeoffs
Data privacy	Fine-grained access control and function-level auditing and provenance, full GDPR (General Data Protection Regulation) compliance
Performance	Fine-grained scheduling and resource management, new performance models and fairness mechanisms that help reduce resource contention and performance variability

Data-intensive applications	Fine-grained data-centric programming models
API jungle	Service discovery and brokering, fine-grained software lifecycle management, standardized multicloud APIs, interoperability, portability, multimodal services

First, the fine granularity for expressing computation adds significant overhead to the resource management and scheduling layers. To overcome this, we envision significant research efforts invested in coscheduling and orchestration for workflows of functions. Moreover, from a user perspective, we need new tools for navigating the cost–performance tradeoffs to explore the complexity of fine-grained pricing models.

Second, data privacy is important for the clients and nontrivial for the providers to offer—for example, ensuring full GDPR (General Data Protection Regulation; https://www.eugdpr.org) compliance. With its fine-grained nature, serverless computing allows for more enhanced access control, function-level auditing, and provenance for seamless, efficient GDPR compliance.

Third, in modern clouds, performance suffers from significant variability due to resource contention, virtualization, and congestion overhead. These issues are amplified in serverless computing, because of its granular nature. However, the increased insight and control over the operational lifecycle provides cloud providers with opportunities to minimize these performance issues by being able to more accurately monitor, profile, and schedule these fine-grained services.

Fourth, data-intensive applications are not naturally expressed in the—stateless—FaaS paradigm. We envision the design and implementation of fine-grained, data-centric, serverless programming models. One promising research direction is investigating distributed promises in serverless environments.

Finally, the API jungle generated by the fast-evolving serverless APIs, frameworks, and libraries represents an important obstacle for software lifecycle management and for service discovery and brokering. To overcome this, significant effort must be invested in multicloud API standardization, interoperability, and portability to avoid lock-in and to enable seamless service discovery.

CONCLUSION

Serverless computing is a promising technology, with a burgeoning market already formed around it. By analyzing the computer technology leading to it, we conclude that this model could not have appeared even a decade ago. Instead, it is the result of many incremental advances, spanning diverse domains: from the increasingly granular resource abstractions, to the emergence of abundant amounts of resources available nearly instantly, to the reduction of costs and complexity of distributed applications.

The current serverless technology offers its customers fine billing granularity, detailed insight and control, and the affordable ability to run arbitrary functions on demand. However, this technology has not been demonstrated beyond selected, convenient applications. We have identified several obstacles and opportunities and have argued that industry and academia must work together. Can we make serverless computing available for many, without the drawbacks of the technology and processes underlying physical containerization?

ACKNOWLEDGMENTS

This work is supported by the Dutch project Vidi MagnaData, by the Dutch COMMIT/ program and the COMMIT/ project commissioner, and by generous donations from Oracle Labs.

REFERENCES

1. M. Levinson, *The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger*, Second Edition, Princeton University Press, 2016.
2. G. Pendse, "Cloud Computing: Industry Report and Investment Case," Nasdaq, 22 June 2017; http://business.nasdaq.com/marketinsite/2017/Cloud-Computing-Industry-Report-and-Investment-Case.html.
3. *Uptake of Cloud in Europe. Digital Agenda for Europe report.*, European Commission, Publications Office of the European Union, Luxembourg, September 2014.
4. *State of the Cloud 2018*, Cloudability, 2018.
5. "Function-as-a-Service Market by User Type (Developer-Centric and Operator-Centric), Application (Web & Mobile Based, Research & Academic), Service Type, Deployment Model, Organization Size, Industry Vertical, and Region - Global Forecast to 2021," Markets and Markets, February 2017; https://www.marketsandmarkets.com/Market-Reports/function-as-a-service-market-127202409.html.
6. E. van Eyk et al., "The SPEC cloud group's research vision on FaaS and serverless architectures," *Proceedings of the 2nd International Workshop on Serverless Computing* (WoSC 17), 2017, pp. 1–4.
7. S. Fouladi et al., "Encoding, fast and slow: Low-latency video processing using thousands of tiny threads," *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation* (NSDI 17), 2017, pp. 363–376.
8. G.J. Popek and R.P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Communications of the ACM*, vol. 17, no. 7, July 1974, pp. 412–421.
9. J. McCarthy, "Recursive functions of symbolic expressions and their computation by machine," *Communications of the ACM*, vol. 3, no. 4, April 1960, pp. 184–195.
10. N. Josuttis, *SOA in Practice: The Art of Distributed System Design*, O'Reilly Media, 2007.
11. R. Heinrich et al., "Performance Engineering for Microservices: Research Challenges and Directions," *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion* (ICPE 17 Companion), 2017, pp. 223–226.
12. E. Dijkstra, "Cooperating Sequential Processes," dissertation, Department of Mathematics, Eindhoven Technological University, 1965.
13. C. Hewitt, B. Bishop, and R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," *Proceedings of the 3rd international joint conference on Artificial intelligence*, vol. IJCAI 73, 2013, pp. 235–245.
14. N. Russell, W.M.P. van der Aalst, and A.H.M. ter Hofstede, *Workflow Patterns: The Definitive Guide*, MIT Press, 2016; https://mitpress.mit.edu/books/workflow-patterns.
15. Survey of the CNCF serverless WG, GitHub, 2018; https://github.com/cncf/wg-serverless.
16. S. Hendrickson et al., "Serverless computation with open-lambda," *Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing* (HotCloud 16), 2016, pp. 33–39.
17. E. Jonas et al., "Occupy the cloud: Distributed computing for the 99%," *2017 Symposium on Cloud Computing*, 2017, pp. 445–451.
18. E. van Eyk et al., "A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures.," *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering* (ICPE 18), 2018, pp. 21–24.

ABOUT THE AUTHORS

Erwin van Eyk is an MSc student in computer science at the Delft University of Technology (TU Delft), where he works on serverless function and workflow scheduling. Van Eyk received a BSc in computer science from TU Delft. He leads the Serverless working group of the Standard Performance Evaluation Corporation RG Cloud Group. Contact him at e.vaneyk@atlarge-research.com; https://erwinvaneyk.nl.

Lucian Toader is an MSc student at Vrije Universiteit Amsterdam, where he studies modern distributed systems. His work on massivizing computer systems led him to serverless. Toader received a BSc in electronics and telecommunication from Politehnica University of Bucharest. Contact him at l.toader@atlarge-research.com.

Sacheendra Talluri is an MSc student in computer science at the Delft University of Technology. In the spring of 2018, he was a research intern at Databricks, working on resource management and scheduling across the memory-storage stack. Talluri received a BTech in information and communication technology from the Dhirubhai Ambani Institute of Information and Communication Technology. Contact him at s.talluri@atlarge-research.com.

Laurens Versluis is a PhD student at Vrije Universiteit Amsterdam, where he studies modern distributed systems. His work on massivizing computer systems focuses on resource management and scheduling, with applications in cloud computing. Versluis received an MSc in computer science from the Delft University of Technology. Contact him at l.f.d.versluis@vu.nl.

Alexandru Uță is a postdoctoral researcher at Vrije Universiteit Amsterdam, where he studies modern distributed systems. His work on massivizing computer systems focuses on resource management and scheduling, with applications in cloud computing and big data. Uță received a PhD in computer science from Vrije Universiteit Amsterdam. Contact him at a.uta@vu.nl.

Alexandru Iosup is a full professor and the University Research Chair at Vrije Universiteit Amsterdam, where he leads the Massivizing Computer Systems group. He's also an associate professor in the Distributed Systems group at the Delft University of Technology. Iosup received a PhD in computer science from the Delft University of Technology. He has received the Netherlands ICT Researcher of the Year award, Netherlands Teacher of the Year award, and several Standard Performance Evaluation Corporation SPECtacular community awards. He's a member of the Young Academy of the Royal Academy of Arts and Sciences of the Netherlands. He's the elected chair of the SPEC Research Cloud Group. Contact him at a.iosup@vu.nl.

Contact department editor George Pallis at gpallis@cs.ucy.ac.cy.

Response to "Scale Up or Scale Out for Graph Processing"

Semih Salihoglu
University of Waterloo

M. Tamer Özsu
University of Waterloo

Editor:
Jimmy Lin
jimmylin@umd.edu

In this article, the authors provide their views on whether organizations should scale up or scale out their graph computations. This question was explored in a previous installment of this column by Jimmy Lin, where he made a case for scale-up through several examples. In response, the authors discuss three cases for scale-out.

Our colleague Jimmy Lin in the University of Waterloo's Data Systems Group wrote an article for this department giving his perspective on whether organizations should scale up or scale out for graph analytics.[1] Similarly to that article, for rhetorical convenience, we use "scale up" to refer to using software running on multicore large-memory machines and "scale out" to refer to using distributed software running on multiple machines.

It is difficult to disagree with the central message of Jimmy's article: For many organizations that have large-scale graphs and want to run analytical computations, using a multicore single machine with a lot of RAM is a better option than a distributed cluster because single-machine software, compared to distributed software, is easier to develop in-house or use out of the box, is often more efficient, and is easier to maintain. This is indeed true, and for the social-network graphs and the computations discussed in that article—e.g., a search for a diamond structure or an online random-walk computation for recommendations—scale-up is likely the better approach. However, Jimmy's article gave the impression that only a handful of applications require scale-out computing, and it failed to highlight several common scenarios in which scale-out is necessary.

In this response article, we discuss three cases for scale-out:

- *Trillion-edge-size graphs*. Several application domains, such as finance, retail, e-commerce, telecommunications, and scientific computing, have naturally appearing graphs at the trillion-edge-size scale.

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

- *Very large extracted graphs*. Graphs are often views extracted from other data sources. Even if the original data sources can fit in the memory of a single machine, the extracted graphs can be orders of magnitude larger, requiring processing on multiple machines.
- *Complex graph computations*. Graph analytics does not comprise random walks, PageRank, connected components, and single-source shortest-path computations, which the database community obsesses about. There are significantly more complex computations that necessitate scale-out even when running on graphs that can fit on a single machine.

In the remainder of this article, we discuss these points and several others, highlighting when scale-out would be necessary. In addition, together with Jimmy, we recently ran a survey of users of graphs (led by our student Siddhartha Sahu), asking them about their graphs, graph software, and graph computations.[2] We have a different interpretation of the survey results than were given in Jimmy's article. We discuss our perspective at the end of this article.

Before we start, let's briefly discuss some terminology. The terms "graph computations" and "graph analytics" are so broadly used that they give very little information about the nature of the computations people refer to with these terms. We will not suggest a fix to this confusion. Often, the meaning will become clear in context, and we will give concrete example computations for the analytics we refer to. As in Jimmy's article, we will, however, ignore OLTP-like workloads, such as "return the two-degree neighbors of node *v*"-type queries that might be issued by user-facing online applications.

PLENTY OF GRAPHS AT THE TOP

Jimmy's article used the trillion-edge scale as the size that justifies scale-out solutions, argued that there are only a handful of such graphs, and asked "How many trillion-edge graphs are there?" Naturally, we should expect power-law phenomena in the number of institutions and organizations and the size of the graphs they have. So, trillion-edge graphs will be significantly fewer than 100-billion-edge graphs, which will be significantly fewer than 10-billion-edge graphs, etc. However, contrary to the common belief, there is not just a handful of trillion-edge graphs. We discuss two commonly appearing examples.

Edge-per-Transaction Graphs

In the database community, large graphs are traditionally associated with the web, the Semantic Web, and social-network graphs from Facebook or Twitter. This limited view fosters the misperception that there are only a few large graphs in the world. Instead, we are observing that large graphs are appearing in domains such as finance, retail, e-commerce, and telecommunications, where an edge is often a transaction or an interaction between entities. We refer to these as *transaction graphs*.

For example, a user makes a transfer from bank account A to bank account B through mobile phone P. This leads to a new transfer edge between A and B and an accessed-via edge between A and P, with other metadata, such as dates or amounts, on these edges. Another example, from retail, is a user going to a retailer's branch and buying a product using a specific credit card, which generates several edges between the entities involved.

Transaction graphs are not limited to finance and retail; they also appear in e-commerce and telecommunications—e.g., a customer reviewing a product or two phones calling each other. When transactions and interactions are modeled as edges between entities, graphs easily reach the trillion scale. This was one of the surprising outcomes of our survey, in which users told us that their most common graphs were transaction graphs consisting of customers, orders, products, and transactions.

Computations for fraud detection and risk and revenue estimation seem to be common on transaction graphs. We are aware of a financial company that has graphs with tens of trillions of edges and is continuously looking for complex fraud patterns on them. One such pattern contains a cycle with eight edges. We also know of an online-ad company that ingests one billion edges

per day for a graph containing nodes of users, URLs, IP addresses, and devices and interaction edges between them. They run proprietary graph algorithms on the graph. Retailers look for recurring purchases and chains of transactions in transaction graphs for revenue estimation. PayPal runs (or used to run) a belief-propagation-like algorithm on its transaction graph[3] on Apache Giraph (http://giraph.apache.org) to compute the risks of its customers and transactions.

It is not difficult to imagine a long list of algorithms and computations organizations run on these very large-scale graphs. At the trillion scale, even without any metadata on the nodes and edges, a scale-out solution is the only reasonable approach.

Scientific-Computing Graphs

There is another scale of graphs in the high-performance computing (HPC) field for scientific computing. This community works on architectures, systems, and algorithms for supercomputers, such as those produced by Cray. We are not in this field, so our knowledge about the graphs here is incomplete. However, we are familiar with some of their publications, benchmarks, and challenges, in which trillion-edge graphs are the norm.

For example, the famous Graph500 benchmark (https://graph500.org) outlines six scales of input graphs on its home page, the largest of which contains 4 trillion nodes and 64 trillion edges. Numerous papers have appeared in HPC venues such as the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), in which graph algorithms are designed for such large-scale benchmarks and are naturally evaluated on tens of thousands of machines. The largest real-world graphs we are aware of from this domain are biological graphs with sizes ranging from tens of billions of edges to a trillion edges (e.g., the human brain's connections).[4–6] As we discuss later, it's not just the size of these graphs that requires scaling out. Sometimes the computational requirements of the applications require scaling out when processing graphs from the HPC domain.

Trillion-size graphs are not limited to transaction and scientific graphs. The US National Security Agency has a "Big Graph Experiment" for processing graphs with tens of trillions of edges.[7] DARPA's HIVE (Hierarchical Identify Verify Exploit) Challenge (https://graphchallenge.mit.edu/darpa-hive) also targets processing trillion-scale graphs. We did not dare ask, but who knows what kind of graphs these institutions have and what they do with them?

EXTRACTING MORE GRAPHS AT THE TOP

Graphs are often extracted from other data sources in practice. Transaction graphs are perhaps the most obvious example of this. Take, for example, PayPal's graph data pipeline described in its Apache Giraph presentation.[3] In this pipeline, raw transaction data is put into a Hadoop cluster and then extracted into a graph model in several formats such as JSON (JavaScript Object Notation) and Gson.

Amol Deshpande observed that it is common to extract multiple graphs views on top of the same raw data and that these views can be orders of magnitude larger than the base data sources.[8] The running example in Deshpande's paper is the DBLP dataset containing papers and their authors. The most obvious graph you can extract from this dataset is a bipartite graph of paper and author nodes and author-wrote-paper edges. Another graph you can extract is a coauthor graph containing an edge between two authors who have coauthored a paper.

As shown in Deshpande's paper, sometimes these extracted graphs can be one or two orders of magnitude larger than the base data. For example, even if a retailer has a transaction graph of its customers and products with one billion edges, a co-purchasing graph that contains an edge between two customers that have purchased the same product can have tens of trillions of edges, requiring scale-out software for processing. We suspect that if the existing scale-out software were scalable enough, users would extract ever larger and more complex graphs from their data sources.

RAW-DATA VERSUS PROCESSABLE-DATA SIZE

While we are still discussing the size of graphs, it is worth noting that the arguments about graph data being small seems to be centered around the size of the raw datasets. Each graph system has its own format for input data, and these input file sizes might not be large. However, to process this data, systems often create internal data structures (e.g., adjacency lists) and various indexes.

We conducted an experiment in which we took a number of real graphs and loaded them to a system—in our case, PowerLyra.[9] The results are shown in Table 1.

Table 1. Processable graph sizes.

Dataset	No. of vertices	No. of edges	Raw data	PowerLyra (single machine)
Live Journal	4.8 M	68.9 M	1.08 Gbytes	6.30 Gbytes
USA Road	23.9 M	58.3 M	951.00 Mbytes	9.09 Gbytes
Twitter	41.6 M	1.4 B	26.00 Gbytes	128.00 Gbytes
UK0705	82.2 M	2.8 B	48.00 Gbytes	247.00 Gbytes
World Road	682.4 M	717.0 M	15.00 Gbytes	194.00 Gbytes

It is important to note the "blow-up" factor in going from raw data to processable data. It is certainly true that the factor is dependent on system implementations and varies among systems. The blow-up might be small for in-house software tailored for a few specific applications, but it is often larger for software that supports a large set of applications. As a simple example, it is common for graph software to index the edges of an input graph both in the forward and backward directions. Once the internal structures of the software are populated, the size of the data that needs to be kept in memory quickly increases. With careful engineering, it might be possible to reduce the amount of expansion, but it is doubtful whether it can be eliminated.

PLENTY OF COMPUTATIONS AT THE TOP

Sometimes, it is not the size of the graphs but the heavy computations running on them that necessitates a scale-out solution. In the database community, we might have over-studied and over-optimized three algorithms: PageRank, the connected-components algorithm based on spreading vertex IDs, and the Bellman–Ford-like single-source shortest-paths algorithm. There are, however, many other algorithms that require significantly more processing than these.

Take, for example, the Markov clustering algorithm for finding clusters in graphs.[10] The algorithm is based on simulating long random walks in the graph. In the linear-algebra framework, it is implemented by iteratively squaring a stochastic matrix of the graph that is obtained initially from the adjacency list matrix. The baseline algorithm requires space and runtime that are quadratic in the number of nodes, and even its optimized versions are computationally expensive. For example, a distributed implementation of the algorithm on a 70-billion-edge graph is reported to take 2.4 hours on a cluster with 2,000 machines.[4]

Another example from scientific computing is de novo genome assembly of de Brujin graphs of small genome segments. In one paper, this was done for the human genome in 8.4 minutes on a cluster with 15,000 machines.[5] There are a lot of such computationally expensive algorithms for tasks such as partitioning,[11,12] node classification,[13] link prediction (e.g., those using the Katz measure of node similarity),[14] and community detection,[15] which would require scale-out implementations. At least, scaling out would allow you to move up one level in the scale of graphs you can process.

In the database community, we do not know of users of graph-processing systems discussing these algorithms, possibly because they are complex and users would run into scalability problems. However, we don't see any reason our community's users would not adopt them if these algorithms came prepackaged in some scalable distributed software.

Another set of computation-heavy algorithms arise in machine learning. There is a growing interest in embedding nodes of graphs into high-dimensional vectors and running machine-learning algorithms on them.[16] We are not experts in this area, but it is not difficult to imagine users wanting to run complex machine-learning algorithms on these representations—say, training deep neural networks with many layers—that likely would require scale-out processing.

WHAT THE SURVEY REALLY SAYS

Let us next give our own interpretation of the survey results[2] that were discussed in Jimmy's article. In our survey, we asked participants about the size of their graphs in terms of the number of edges. The largest scale we offered them was "1 billion edges and above.'' Twenty of the 89 participants picked this largest scale. These participants were recruited from the email lists of 22 software programs for processing graphs, including graph databases, RDF triple stores, graph analytics libraries, distributed graph engines, and graph visualization software.

Jimmy interpreted these results as evidence that graphs in practice are not very large because storing a billion-edge graph would require 8 Gbytes of RAM. We have a different interpretation of these results, for two reasons.

Our first reaction when we got the results was the realization that we did not do a very good job in formulating the graph size questions. We should have had at least two larger scales, going up to 100-billion-edge graphs. To remedy this problem, we surveyed email lists of the 22 graph software applications in our list and found graphs that were a lot larger than 1 billion edges (see Table 18 in the paper[2]). Specifically, we found 42 users indicating graphs between 1 billion and 10 billion edges, 17 users with 10 billion to 100 billion edges, six users between 100 billion and 500 billion edges, and one user with a graph larger than 500 billion edges. These numbers suggest that at least graphs on the scale of 100 billion edges and above no longer exist only at Google, Facebook, and Twitter and are becoming prevalent.

Second, and maybe more important, when asked about their major problems, more than half of the participants stated that their biggest problem is scalability. This was the single most popular challenge. Here, our guess is that a lot of users are using single-machine software and are having trouble scaling with it. We cannot tell which specific workloads participants are trying to scale, but it is not far-fetched to think that at least some of them are non-OLTP batch computations such as clustering, node centrality computations, or link prediction.

To us, the survey contains more evidence supporting scale-out solutions than evidence supporting scale-up solutions. However, readers should always keep in mind that the survey might not be very representative of users with non-OLTP workloads, which is what we focus on here. This is because we suspect that many (probably most) of the survey participants were users of graph databases, such as Neo4j, which are used mainly for OLTP-like workloads.

CONCLUSION

Let's conclude by revisiting an old question that is destined to come up in any scale-up versus scale-out discussion: how will organizations scale up or scale out when needed? It is possible to implement a completely serial or multithreaded algorithm for a workload, and it is possible that these implementations might perform better than scale-out solutions. However, when the time comes, scaling up often means putting more RAM on an existing machine or buying or renting a larger machine with more RAM or more cores. This might be quite costly, slow, or limited: the next big machine might be on the market or cloud in a few months and only have a few more CPUs. Scaling out by buying or renting more of the existing machines instead will likely be cheaper and simpler. This is about hardware.

On the software side, organizations should rightfully be concerned that existing distributed software will not scale, following the now-famous COST (configuration that outperforms a single

thread) arguments against it.[17] For example, Jimmy's article noted that "many scale-out distributed graph processing frameworks are just terrible in terms of performance.''[1] However, it is important to realize that this is not a fundamental limitation of scale-out software. Certainly not all distributed software is terrible. For example, the Naiad system[18] developed by the authors of COST was demonstrated to have a low COST for graph computations.[17] One thing we know from history is that with more engineering efforts, software will improve in performance, and the COST of existing distributed graph software should get better over time.

The discussion of scale-up versus scale-out is an old one—both in general and as it relates to graph processing. Certainly, if your graph size fits your machine's main memory and single-machine software suits your needs, this should indeed be the first thing to try. However, our research and experience suggest that there are many cases, some of which we outlined in this article, in which this solution is not generally suitable. Another thing we know from history is that data sizes never decrease but continue increasing, so it is important for organizations to anticipate this.

REFERENCES

1. J. Lin, "Scale up or scale out for graph processing?," *IEEE Internet Computing*, vol. 22, no. 3, 2018, pp. 72–78.
2. S. Sahu et al., "The ubiquity of large graphs and surprising challenges of graph processing," *Proceedings of the VLDB Endowment*, vol. 11, no. 4, 2017, pp. 420–431.
3. J. Tang, "Graph Mining With Apache Giraph," 2013; www.slideshare.net/Hadoop_Summit/tang-june26-205pmroom210cv2.
4. A. Azad et al., "HipMCL: A High-performance Parallel Implementation of the Markov Clustering Algorithm for Large-scale Networks," *Nucleic Acids Research*, vol. 46, no. 6, 2018.
5. E. Georganas et al., "Hipmer: An extreme-scale de novo genome assembler," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (SC 15), 2015.
6. C. Zimmer, "100 Trillion Connections," *Scientific American*, vol. 304, no. 1, 2011, pp. 58–63.
7. P. Burkhardt and C. Waring, "An NSA Big Graph Experiment," 2013; www.pdl.cmu.edu/SDI/2013/slides/big_graph_nsa_rd_2013_56002v1.pdf.
8. K. Xirogiannopoulos and A. Deshpande, "Extracting and Analyzing Hidden Graphs from Relational Databases," *Proceedings of the 2017 ACM International Conference on Management of Data* (SIGMOD 17), vol. SIGMOD 17, no. 2017, 2017.
9. R. Chen et al., "PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs," *Proceedings of the Tenth European Conference on Computer Systems* (EuroSys 15), 2015.
10. S. Van Dongen, "Graph Clustering Via a Discrete Uncoupling Process," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, 2008.
11. G. Karypis and V. Kumar, "A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, 1998.
12. A. Pothen, H.D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, 1990.
13. S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node Classification in Social Networks," *Social Network Data Analytics*, C. Aggarwal, Springer, 2011.
14. D. Liben-Nowell and J. Kleinberg, "The Link Prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, 2007.
15. Z. Yang, R. Algesheimer, and C.J. Tessone, "A Comparative Analysis of Community Detection Algorithms on Artificial Networks," *Scientific Reports*, vol. 6, 2016, p. 30750.

16. A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," *Proc. 22nd ACM International Conference on Knowledge Discovery and Data Minin* (KDD 16), 2016, pp. 855–864.
17. F. McSherry, M. Isard, and D.G. Murray, "Scalability! But at what cost?," *Proceedings of the 15th USENIX conference on Hot Topics in Operating Systems* (HOTOS 15), 2015.
18. D.G. Murray et al., "Naiad: A Timely Dataflow System," *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (SOSP 13), 2013.

ABOUT THE AUTHORS

Semih Salihoglu is an assistant professor at the University of Waterloo's Cheriton School of Computer Science. His research focuses on new theories, algorithms, and systems for processing and managing graph data. Salihoglu received a PhD in computer science from Stanford University. Contact him at semih.salihoglu@uwaterloo.ca.

M. Tamer Özsu is a professor at the University of Waterloo's Cheriton School of Computer Science. His research interests involve the application of database technology to non-traditional data types, and distributed and parallel data management. Özsu received a PhD in computer and information science from the Ohio State University. Contact him at tamer.ozsu@uwaterloo.ca.

Contact department editor Jimmy Lin at jimmylin@umd.edu.

To Follow or Not to Follow: A Study of User Motivations around Cybersecurity Advice

Michael Fagan
University of Connecticut

Mohammad Maifi Hasan Khan
University of Connecticut

Usable-security researchers have long pondered what motivates some users to ignore advice and make decisions that appear to put their security and privacy at risk. The study reported here specifically investigated user motivations to follow or not follow computer security advice, through a survey distributed via Amazon Mechanical Turk. To guide the study design, the authors used a rational-decision model and current thought on human motivation. The data shows key gaps in perception between those who followed the tested pieces of advice (update software, use a password manager, use two-factor authentication, or change passwords) and those who did not and helps explain the participants' motivations behind their decisions. Notably, the study found that social considerations were broadly trumped by individualized rationales.

User perceptions and adoption of various security tools and techniques has remained a popular research topic. Some studies have identified[1–2] and others have attempted to explain[3–6] a divergence between recommended actions and actual protections used by the public. This question of why some people follow security advice, while others do not, has been touched on before. However, to the best of our knowledge, it has not been broadly approached using empirical data collected and analyzed for that purpose.

We investigated the motivations of users to follow or not follow common computer security advice. Users' decision making was modeled using a cost–benefit framework, with the concepts of risk and social motivation added. On the basis of this investigation, we present the findings from a web-based survey constituting both qualitative and quantitative data that was distributed to and completed by 290 Amazon Mechanical Turk users.

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

As a foundation for our survey, we used four common security recommendations: updating software, using a password manager, using two-factor authentication (2FA), and changing passwords. For each piece of advice, we formed two types of groups: those who followed the advice (Yes groups) and those who did not (No groups). We then compared their perceptions. Before collecting and analyzing data, we hypothesized that the benefits, costs, and risks of following and not following each piece of advice would be rated in a way that agreed with the participants' reported decision to follow or not follow the advice. Additionally, we expected that social motivations would be rated lower than individual motivations.

These hypotheses were largely correct. The benefit of not following was rated higher by those who didn't follow each piece of advice, whereas the risk of not following was rated higher by those who did follow each piece of advice. Furthermore, the cost of not following was seen as higher by those who followed each piece of advice compared to those who did not, for all the pieces of advice, except using 2FA. These findings indicate that each group viewed its decisions as the rational one, as we expected. Finally, individual concerns were rated higher than social concerns for all variables, indicating low social motivation around computer security.

BACKGROUND

Although complex, human decision making can be viewed as a consideration of cost and benefit, in which humans are rational actors who choose to minimize cost and/or maximize benefit. Herley highlighted this model in his work exploring the motivations around following security advice, citing the low chance of a security breach for any given user (representing low benefit) and the high cost of daily security maintenance.[4] He also suggested that more data is needed to determine the actual cost and benefit of these decisions to better inform the advice experts give. Another study showed that users show rational tendencies when considering whether to accept advice, such as looking to the trustworthiness of the source in some cases (e.g., antivirus software), but relying on personal evaluation in others (e.g., passwords).[7] Inspired by these prior efforts, we used a cost–benefit framework as the starting point for our study's design.

In addition to cost and benefit, the literature shows us that risk perception is central to security-related behavior.[5,8] Literature surveys have identified security risks and risk perceptions as key considerations in many studies that focus on psychology and computer security.[5] The study designs of recent usable-security research have focused on risk as well.[2,9] Thus, we incorporated perceptions of risk along with cost and benefit.

Although risk perception is intrinsically linked with security decisions, we also added social motivations (motivations driven by values or wanting to help or please others), which are argued to be independent of and much stronger or longer lasting than instrumental motivations (motivations related to gaining material reward or avoiding material cost).[10] Some researchers have investigated social motivations in the area of usable security. For example, Das et al. found that users could be better motivated to act securely online if their peers would know the decisions they were making.[11] Therefore, our study's model was expanded to include how participant users thought their decisions affected users of other computers.

Finally, there is evidence in the literature suggesting that experts and average users think and act differently when it comes to computer security. Ion et al. showed that experts and regular users reported different behaviors when asked which they think are the best for staying safe.[1] Additionally, another study found that security-sensitive users (as many experts arguably are, on the basis of their security-conscious behavior reported in Ion et al.'s study) and general users differ in their sources of security advice.[12] However, Kang et al. found that there was no direct correlation between participants' technical background and the actions they took to control their privacy.[2] Thus, rather than separating users into experts and nonexperts, we simply compared those who followed each piece of advice with those who did not, to identify differences in perceptions between those two groups.

METHODS

As the previous section described, our study was designed to investigate the motivations of users to follow or not follow common computer security advice, using several cues from prior studies. We focused on four pieces of common security advice harvested from Ion et al.'s 2015 paper:[1]

- keeping software up to date,
- using a password manager,
- using 2FA, and
- changing passwords frequently.

For each, we sampled two groups of users: those that followed the advice and those that did not. To help in describing the study, we refer to the samples of users who followed each piece of advice as Yes groups, whereas we refer to the samples of users who did not follow the advice as No groups. All groups were sent a similarly designed survey to gauge their motivations.

Survey Content

We extended the traditional cost–benefit analysis to include the perception of risk and consider the social aspect of each decision. These concepts were formalized into the following 12 variables:

1. Individual Benefit of Following
2. Social Benefit of Following
3. Individual Cost or Inconvenience of Following
4. Social Cost or Inconvenience of Following
5. Individual Risk of Following
6. Social Risk of Following
7. Individual Benefit of Not Following
8. Social Benefit of Not Following
9. Individual Cost or Inconvenience of Not Following
10. Social Cost or Inconvenience of Not Following
11. Individual Risk of Not Following
12. Social Risk of Not Following

Variables were defined using survey instruments phrased differently for each piece of advice, Yes or No group, and specific variable. One of these two phrasings was used to define each variable:

- *Phrasing A*. How much would you say [you | users of other computers] are [benefited | cost or inconvenienced | put at risk] by you (not) [following the advice]?
- *Phrasing B*. How much would you say [you | users of other computers] would be [benefited | cost or inconvenienced | put at risk] if you did (not) [follow the advice]?

Variables 1 through 6 were defined using Phrasing A for the Yes groups and Phrasing B for the No groups. Variables 7 through 12 were defined with Phrasing B for the Yes groups and Phrasing A for the No groups. This was done to match the instrument's phrasing to the participant's reported decision. Individual variables used "you" in the first bracket, whereas social variables used "users of other computers." The second brackets were likewise replaced for the variables that asked about benefit, cost or inconvenience, and risk. Finally, "follow (or following) the advice" was replaced as appropriate for each piece of advice that we tested in the surveys (e.g., "use (or using) 2FA"), with "not" being added as needed.

Because each variable was defined in a slightly different format for the Yes and No groups, our analysis compared ratings that were more or less hypothetical, depending on the group. The goal of this work was to identify the possible gaps in perceptions between those who followed the security advice and those who did not.

For the decisions examined in this study, because the users might have been pondering a behavior they had not practiced in the past, at least some of their considerations might have been hypothetical. Their perceptions of possible outcomes might have been skewed or biased, which we hoped to identify. Therefore, our study had to compare hypothetical ratings with more grounded reports.

A 4-point Likert scale was used for each of the quantitative questions described above (i.e., 1 = none, 2 = little, 3 = some, and 4 = a lot). An open-ended statement requesting survey takers to indicate why they chose to follow or not follow the target advice ("Please explain in a few sentences why you choose to (not) [follow the advice]") was shown to participants first, on a separate page in all surveys. We did this to avoid biasing the open-ended responses toward our overall study framework.

The quantitative questions were then divided into two additional pages in the survey. The first asked about perceptions of the participant's reported behavior; the second asked about perceptions of the opposite behavior.

Survey templates for both groups, showing the format, are in the appendix of our 2016 SOUPS (Symp. Usable Privacy and Security) paper.[3]

Sampling Methodology

We used Mechanical Turk to gather an initial sample of participants who answered a screening survey that asked for basic demographic information as well as a report of which of our study's advice they did or did not follow. Participants were compensated $0.25 for completion of these instruments. This compensation level was set low owing to the small number of instruments included. Participants were informed that they could be contacted with an additional survey based on their responses to this initial set of instruments, but no indication was given as to how eligibility would be determined.

On the basis of the screening survey responses, we formed the Yes and No group for each piece of advice by randomly selecting 50 participants that matched the group's target behavior. For example, the Yes group for updating comprised a random selection of 50 respondents who said they updated on the initial survey. A participant selected to be in a group was not considered for inclusion in others.

Follow-up participants were contacted with the appropriate survey through Mechanical Turk's messaging system. They were informed that their continued participation was entirely voluntary. If they chose to continue, they were compensated another $4 for their time and effort on the longer survey. Not all 50 for each group replied, resulting in samples between 30 and 40 participants for each, for a total of 290 across all eight groups.

EVALUATION

Our analysis was guided by our study model framework. We identified gaps in perceptions between the Yes and No groups for each piece of advice, with particular divergence around the benefit and risk associated with their decisions. Our qualitative data helped us understand some possible explanations for the differences between groups. Finally, we revisited the quantitative data to highlight the dominance of individual considerations around deciding to follow computer security advice.

Gaps in Perceptions

Before data collection, we hypothesized that the participants' ratings would align with their reported behavior. Those who didn't follow each piece of advice were expected to rate the risk and cost of doing so as lower than those who did follow the advice, while also rating the benefit higher. Conversely, those who did follow the advice were each expected to rate the benefit as

higher than those who didn't follow the advice, while rating the risk and cost of doing so as lower.

Through analysis of the data, we saw significant gaps in the rated benefit, cost, and risk involved in the decision to follow each piece of advice, when comparing the Yes and No groups that generally agreed with our hypothesis. Table 1 shows summaries of responses for each variable, from each piece of advice's Yes and No groups, along with Mann-Whitney U tests comparing the distribution.[13] We used Cohen's *d* to express the effect size of the difference between the Yes and No groups' distributions.

Table 1. Rating summaries for all variables, from each group, with U tests comparing the distribution between each Yes group (participants who followed the advice) and No group (participants who did not follow the advice). The effect size is measured with Cohen's *d*. A shaded background indicates results that are of significance $p < 0.004$.

		Of following					*Of not following*				
		Yes	No	U test			Yes	No	U test		
		Avg. (Mdn.)	*Avg. (Mdn.)*	*U*	*p*	*d*	*Avg. (Mdn.)*	*Avg. (Mdn.)*	*U*	*p*	*d*
Individual benefit	Update	3.77 (4)	2.97 (3)	275	<0.001	0.5	1.51 (1)	2.13 (2)	348	0.002	0.4
	PW mgr.	3.78 (4)	2.50 (2.5)	155	<0.001	0.7	1.68 (1)	2.70 (3)	302	<0.001	0.5
	2FA	3.71 (4)	2.90 (3)	244	<0.001	0.5	1.59 (1.5)	2.62 (3)	162	<0.001	0.6
	Change PW	3.47 (4)	2.53 (3)	256	<0.001	0.6	1.70 (2)	3.03 (3)	176	<0.001	0.7
Social benefit	Update	2.71 (3)	2.39 (3)	338	0.286	0.1	1.40 (1)	1.58 (1)	371	0.371	0.1
	PW mgr.	2.08 (2)	1.70 (1)	499	0.155	0.2	1.39 (1)	1.68 (1)	511	0.142	0.2
	2FA	2.48 (2)	2.29 (2)	390	0.489	0.1	1.59 (1)	1.92 (1.5)	314	0.237	0.2
	Change PW	1.73 (1)	1.48 (1)	464	0.235	0.2	1.74 (1)	1.58 (1)	511	0.467	0.1
Individual risk	Update	1.56 (2)	1.72 (2	497	0.335	0.1	3.42 (4)	2.77 (3)	337	0.002	0.4
	PW mgr.	1.83 (2)	2.53 (2)	343	<0.001	0.5	2.88 (3)	1.80 (2)	303	<0.001	0.5
	2FA	1.56 (1)	1.62 (1)	499	0.729	0	3.42 (3)	2.61 (3)	244	<0.001	0.5
	Change PW	1.35 (1)	1.71 (2)	499	0.014	0.3	3.14 (3)	2.63 (3)	441	0.003	0.3
Social risk	Update	1.13 (1)	1.38 (1)	370	0.047	0.3	2.67 (3)	1.76 (1)	263	<0.001	0.4
	PW mgr.	1.41 (1)	1.53 (1)	628	0.707	0	1.92(2)	1.29 (1)	409	0.002	0.4
	2FA	1.31 (1)	1.48 (1)	434	0.47	0.1	2.48 (3)	1.79 (2)	289	0.013	0.3
	Change PW	1.19 (1)	1.17 (1)	629	0.709	0	1.70 (1)	1.29 (1)	483	0.044	0.2

Individual cost	Update	2.03 (2)	2.1 (2)	528	0.444	0.1	2.95 (3)	2.00 (2)	248	<0.001	0.5
	PW mgr.	1.73 (2)	2.18 (2)	533	0.011	0.3	3.15 (3)	1.75 (1)	245	<0.001	0.6
	2FA	2.00 (2)	2.39 (2)	406	0.036	0.3	1.76 (1)	1.57 (1)	447	0.451	0.1
	Change PW	2.35 (2)	2.97 (3)	450	0.005	0.3	2.28 (3)	1.61 (1)	426	0.003	0.4
Social cost	Update	1.22 (1)	1.29 (1)	431	0.781	0	2.32 (2)	1.59 (1)	248	0.001	0.4
	PW mgr.	1.28 (1)	1.52 (1)	566	0.213	0.2	1.84 (1)	1.03 (1)	354	<0.001	0.5
	2FA	1.52 (1)	1.44 (1)	404	0.786	0	1.69 (1)	1.41 (1)	343	0.356	0.1
	Change PW	1.28 (1)	1.65 (1)	491	0.073	0.2	1.50 (1)	1.24 (1)	526	0.174	0.2

BENEFIT

Benefit can be a key motivator if a decision appears to provide it. Our study measured both the rated benefit of following and not following each piece of advice. By comparing these benefit ratings between the Yes and No groups, we could see whether the participants diverged in their perceptions of the benefit related to each piece of advice.

As seen in Table 1, for all pieces of advice, the Yes groups rated the benefit of following the advice as significantly higher than the No groups rated the benefit they thought they would get if they followed the advice. Social Benefit of Following was not rated significantly differently between groups for any piece of advice.

We found a similar, but mirrored, tale for Individual Benefit of Not Following. In this case, the No groups rated their benefit as significantly higher than the Yes groups rated the benefit they would experience if they no longer followed the advice. Again, there were no significant differences between groups on any piece of advice for Social Benefit of Not Following.

When we looked at the qualitative data to help explain this gap, a few patterns emerged across the pieces of advice. About half of those who said they updated frequently and half who said they used a password manager mentioned the added security benefit as a reason for their decision. Furthermore, 72% ($N = 36$) of those who used 2FA and 86% ($N = 37$) of those who frequently changed their password mentioned security benefits in their comments. Some decisions seemed to have other benefits as well, such as getting the latest software through updating (10 of the 39 participants who updated regularly) or the added convenience of a password manager (37 of the 40 password manager users).

Risk

Decisions related to security are uniquely tied to risk. So, we dug into the participants' ratings of the risk involved in and avoided by their security decisions, to better understand their thinking. Looking at Table 1, for all or most advice, Individual Risk of Not Following and Social Risk of Not Following were rated lower by the No groups than by the Yes groups.

Many comments from the No groups mentioned a lack of worry about the risk as a reason for their decision. About a fifth of comments from both those who did not use 2FA (19%) and did not change passwords regularly (18%) said they decided not to follow the advice because they didn't care if they were hacked.

Sometimes, rather than not caring about a risk, participants in the No groups expressed heightened attention to the perceived risk, which impacted their decision. For example, eight comments from those who did not update said they wanted to avoid change or harm. Notably, almost half (45%, $N = 38$) of those who did not use a password manager said they were explicitly avoiding a security risk they saw in using such a tool. We also see this sentiment in Table 1, where the password manager No group rated Individual Risk of Following as significantly higher than the Yes group. In fact, Individual Risk of Following was rated significantly differently between the Yes and No groups only for using a password manager; no other advice had significant differences for this variable.

The No group's qualitative comments shed some light. Twelve of the 38 comments mentioned avoiding centralization of passwords as a reason for deciding to not use a password manager. The Yes groups showed a pattern of being motivated by a security benefit or avoidance of risk; ironically, many from the password manager No group were similarly motivated in their counter-decision.

Cost

No matter how prominent the risk being avoided, if the cost is too high, many users will refuse or be unable to follow the advice. Thus, we analyzed how participants from each group rated the cost of following and not following each piece of advice.

For updating frequently and using a password manager, the Yes groups rated Cost of Not Following as higher than the No groups for both the individual and social phrasings. Additionally, the individual phrasing was rated significantly differently for changing passwords. Over half (56%, $N = 39$) of the comments from those who updated reported avoiding bugs as a reason for their action. The large number of comments from the users of password managers that mentioned the convenience of the tool showed how advice-specific implications can alter decision making for some users.

Cost was a top complaint from the No groups. Almost a quarter (23%, $N = 30$) of those who didn't update, half (48%, $N = 31$) of those who didn't use 2FA, and over half (53%, $N = 38$) of those who didn't frequently change passwords said the avoidance of an inconvenience influenced their decision. For not using 2FA, 23% ($N = 31$) mentioned avoiding a cost explicitly, while 39% ($N = 38$) of those who did not frequently change passwords said they made their decision because frequently changing passwords would be hard to remember to do or make their passwords hard to remember. These findings highlight how the context of a decision, specifically the cost involved, might have a big impact on a user's action.

Social versus Individual Motivations

Although many comments from the participants at the start of the survey conformed to the benefit–risk–cost breakdown in our study's framework, the social–individual divide was not as apparent. Only 13 of the 290 comments we received mentioned any kind of social motivation, all from the Yes groups. Prompted by this and our study hypothesis that the individual ratings would be higher than the social ratings, we used the quantitative data to investigate further.

Figure 1 shows the relative magnitude of the ratings for the individual phrasings of each variable compared to the social phrasings. The individual phrasings were rated consistently higher for all, regardless of the advice or decision. Statistical tests of these differences showed that they were significant.

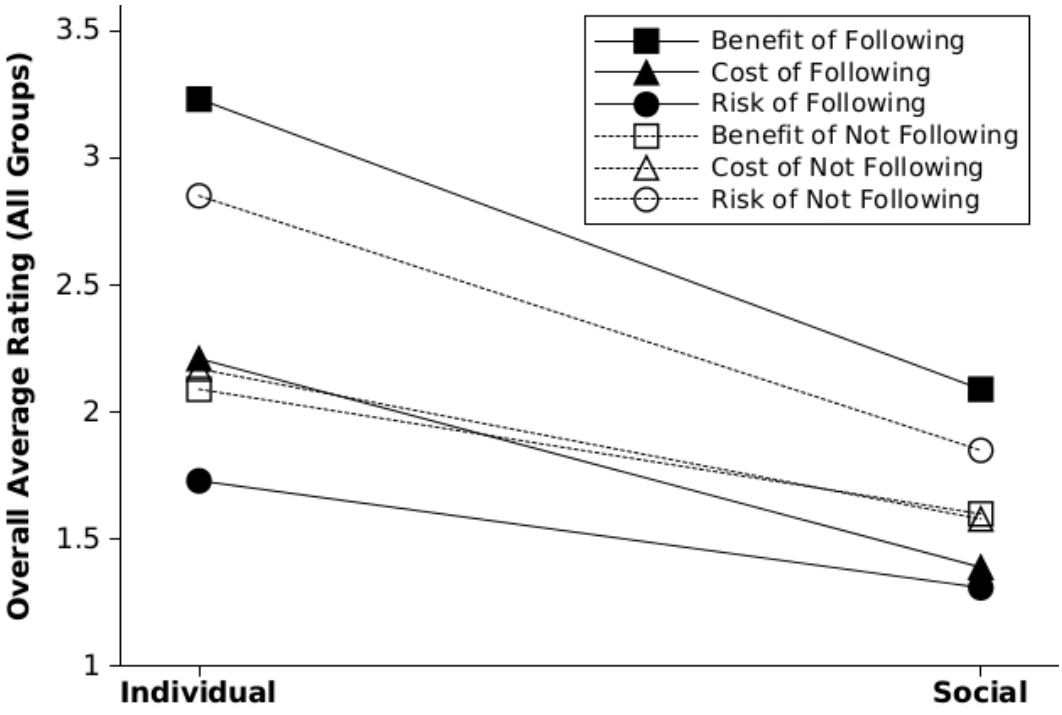

Figure 1. A plot of the average overall ratings for each variable, arranged to show the consistently lower social ratings compared to the individual ratings. A sign test of each variable pair (individual versus social) found significant ($p < 0.001$) differences for all variables.

DISCUSSION

Participants in our study generally rated the benefit, risk, and cost around each decision in a way that agreed with their reported behavior, which we expected in our hypotheses. For example, for all advice, Benefit of Following was rated higher for the Yes groups than the No groups, while Benefit of Not Following was rated higher by the No groups. This generally followed for risk and cost as well, where the Yes groups were apt to rate the risk and cost avoided by their decision as higher than the No group. This trend should be unsurprising, as you would expect an adherent to think he or she is getting more benefit (or avoiding more cost and/or risk) than a nonadherent. However, it is important to keep in mind that those you are advising might have a different outlook than you. So, it is imperative for the advice experts give to be provably effective and usable because they must be convincing when trying to motivate new behavior.

Notably, for most (but not all) advice, Individual Costs of Following and Individual Risks of Following were not rated significantly differently between the Yes and No groups. This could indicate that the groups agreed on the cost and risk of following those pieces of advice. This evidence adds credence to calls from others that at least some users ignore security advice because of high cost and/or low benefit.[4] In our sample, benefit was generally rated differently, whereas cost was rated much more similarly between the Yes and No groups. Although more data is certainly needed, it could be that some who do not adhere to good security practice do not see a usability issue in following the advice (as indicated by their agreement with the Yes groups on cost), but have a different perspective on the risk of their inaction and/or the efficacy of solutions.

Our participants agreed in another way across the Yes and No groups: their motivations were predominately individual rather than social, another result we expected in our hypotheses. We saw this trend in both the quantitative and qualitative data, and it makes sense when you consider the individualized aspects of using a computer. Many times, individuals are physically alone when using their devices, which could impart a sense of isolation, even when on the Internet.

Although the participants did not identify it, their computer security decisions do affect others. For example, if a user chooses not to update, causing his or her device to be compromised and assimilated into a malicious botnet through a security flaw, then that user's decision not to update could impact others when their device participates in a distributed attack. Although newer research has pointed to the power of social motivation,[10] given the lack of social consciousness around security decisions, future work that attempts to motivate users through social motivation needs to rethink how to best approach the problem.

Finally, it is interesting to note that, on the basis of the project from which they were extracted, three pieces of the advice tested in this study were commonly recommended by experts, whereas the fourth (changing passwords frequently) was not commonly recommended by experts but was still regularly cited by average users as a way to stay safe online (i.e., folk advice).[1] Looking at our data, there are some trends that differentiated the folk advice from expert advice. Specifically, frequently changing passwords seemed to offer the least benefit while simultaneously being the costliest, as can be seen in the ratings summaries of Table 1. Interestingly, this trend held for both the Yes and No groups, showing another area in which, despite a divergence in behavior, the participants did not disagree in their ratings.

Identifying why, despite lower efficacy and higher cost, some users still gravitate toward folk advice such as changing passwords, while balking at many expert recommendations, is imperative to understand their decision making in this space. For example, some users might get their security habits from the IT policies of their workplace, which commonly suggest or require regular password changes, highlighting a subtle channel of communication (i.e., corporate IT policy) security experts might be able to utilize to increase secure behavior. Further investigation is needed to better inform these findings.

Our approach is not without limitations. For instance, although we were able to find statistically significant differences in many places, more data from more users could generate additional findings or new insight into existing findings. Larger samples could garner stronger effect sizes than those in this study, which were generally moderate. In addition, examination of more types of advice and contexts (e.g., perceptions of the benefit, risk, and cost of specific kinds of devices) could also broaden the picture. An expanded decision-making framework might provide more insight but would likely require a study larger than that presented here, introducing different limitations. Finally, because Mechanical Turk's population is not representative of the general population, replication of this study with more samples would help generalize the findings.

CONCLUSION

Our results show differences in the perceptions of benefit, risk, and cost associated with the decision to adhere to a variety of recommended security behaviors. Both those who did and did not follow each piece of advice reported that their decision provided them more benefit than if they changed. Those who followed the advice rated the risk of changing their decision as much higher than did those who did not follow the advice. The cost of not following the advice was also seen as higher by those who followed it than by those who did not. Finally, we found that individual concerns were rated consistently higher than social concerns.

More data about the actual risk and cost incurred by users as well as further investigations into perceptions will serve to better frame and explain the findings of this study. Nonetheless, our results have provided insight into user motivation in this context and serve to inform future efforts toward the broader goals of the usable-security field.

ACKNOWLEDGMENTS

This work is supported by the US National Science Foundation under grant CNS-1343766 and by GAAAN (Graduate Assistance in Areas of National Need) Fellowship P200A130153. Any opinions, findings, or recommendations expressed are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

1. I. Ion, R. Reeder, and S. Consolvo, "'...no one can hack my mind:' Comparing expert and non-expert security practices," *Symposium on Usable Privacy and Security* (SOUPS 15), 2015, pp. 327–346.
2. R. Kang et al., "'My data just goes everywhere:' User mental models of the internet and implications for privacy and security," *Symposium on Usable Privacy and Security* (SOUPS 15), 2015, pp. 39–52.
3. M. Fagan and M.M.H. Khan, "Why do they do what they do?: A study of what motivates users to (not) follow computer security advice," *Symposium on Usable Privacy and Security* (SOUPS 16), 2016, pp. 59–75.
4. C. Herley, "So long, and no thanks for the externalities: The rational rejection of security advice by users," *New Security Paradigms Workshop*, 2009, pp. 133–144.
5. A.E. Howe et al., "The psychology of security for the home computer user," *IEEE Symposium on Security and Privacy* (SP 12), 2012, pp. 209–223.
6. G. Stewart and D. Lacey, "Death by a thousand facts: Criticising the technocratic approach to information security awareness," *Information Management & Computer Security*, 2012, pp. 29–38.
7. E.M. Redmiles, S. Kross, and M.L. Mazurek, "'How I Learned to be Secure:' a Census-Representative Survey of Security Advice Sources and Behavior," *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 666–677.
8. F. Asgharpour, D. Liu, and L.J. Camp, "Mental models of security risks," *International Conference on Financial Cryptography and Data Security*, 2007, pp. 367–377.
9. M. Harbach et al., "Using personal examples to improve risk communication for security and privacy decisions," *Annual ACM Conference on Human Factors in Computing Systems* (CHI 14), 2014, pp. 2647–2656.
10. T.R. Tyler, *Why people cooperate: The role of social motivations*, Princeton University Press, 2010.
11. S. Das et al., "The effect of social influence on security sensitivity," *Symposium on Usable Privacy and Security* (SOUPS 14), 2014, pp. 143–157.
12. E.M. Redmiles, A.R. Malone, and M.L. Mazurek, "'I think they're trying to tell me something:' Advice sources and selection for digital security," *IEEE Symposium on Security and Privacy* (SP 16), 2016, pp. 272–288.
13. H.B. Mann and D.R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, 1947, pp. 50–60.

ABOUT THE AUTHORS

Michael Fagan is a recent PhD graduate from the University of Connecticut's Computer Science and Engineering Department. His research interests include the human factors of cybersecurity and adoption of secure behaviors, particularly human motivations when making risk-associated decisions. Fagan received a BA in history and computer science from Vanderbilt University. Contact him at michael.fagan@uconn.edu.

Mohammad Maifi Hasan Khan is an associate professor in the University of Connecticut's Computer Science and Engineering Department. His research interests include usable cybersecurity, risk communication, trust in automation, and performance modeling and troubleshooting of large-scale systems. Khan received a PhD in computer science from the University of Illinois, Urbana-Champaign. Contact him at maifi.khan@uconn.edu.

Real-Time Identity-Deception Detection Techniques for Social Media: Optimizations and Challenges

Michail Tsikerdekis
Western Washington University

Identity-deception detection methods have been proposed for social-media platforms with high effectiveness, but their efficiency can vary. Previous literature has not examined the potential of these methods to work as real-time monitoring systems. Such implementations further highlight the challenges of applying computationally intensive methods in online environments that involve large datasets and high speeds of data. This paper attempts to classify detection methods based on the approaches and identifies factors that, in real-time systems, will impact the effectiveness and efficiency of these methods. Optimizations are proposed that can limit the computational overhead. Further challenges involving real-time identity-deception detection are discussed.

Identity deception in social media has received attention in recent years.[1,2] The rapid growth of the user population for social-media platforms coupled with the ease of creating new accounts has increased identity-deception attacks.[3] These attacks can create disruptions in the normal operations of online communities and impact their users. Research attention to novel identity-deception detection methods has addressed the issue of identity deception using computational solutions. These are more effective than detection performed by humans. However, the computational overhead is rarely taken into account within the size and scope of social-media platforms. These computational solutions can only realistically be expected to have an impact on reducing identity-deception cases if they can monitor online communities on a real-time basis. The longer a deceptive account is allowed to operate unchecked, the larger its negative impact on an online

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

community. The ideal goal is to continuously monitor a new account and render judgments based on its actions until a confidence threshold is surpassed that would allow for classifying it as legitimate or malicious. However, literature has not examined the potential of these detection methods to monitor these online communities in real time.

This paper provides a survey of existing identity-deception detection methods and identifies their potential application as real-time monitoring systems in large-scale social-media platforms. Their performance is discussed based on the estimated time complexity and their ability to cope as the data velocity increases. Optimizations for these and future identity-deception detection techniques are proposed for large-scale social-media platforms that can help mitigate the computational overload when monitoring systems in real time. Given the scope and size limitations of this article, the approach taken is theoretical and aims to open a discussion on novel approaches where real-time detection systems can be made possible for social-media platforms.

IDENTITY-DECEPTION ATTACKS

The social-media identity-deception attacks that this paper considers are identity theft and identify forgery. These involve generating accounts and employing identity management,[4] as well as behavioral strategies (such as navigating to a particular page) aimed at deceiving others. Identity-management strategies are context-specific and begin with the selection of login credentials and subsequent editing of various attributes associated with the account (such as age and employment). Table 1 shows categories and examples of such attacks. Identity creation describes scenarios in which a legitimate identity in real life does not exist in an online platform (such as Facebook), yet the attacker creates the identity based on the data available from other sources.[1] Profile cloning involves cases in which a legitimate identity is present and cloned.[3] Disruption attacks are direct in their nature (such as spamming on a page until banned), and attackers often expect to be caught. Sockpuppetry, on the other hand, involves attacks that are more sophisticated; often, a puppeteer's account is also present in the platform (albeit probably banned).[5] Finally, Sybil attacks are a special case of sockpuppetry with a network-centric focus.[6] Multiple identities are generated towards achieving an often long-term goal.

Table 1. Identity-deception attacks.

Category	Subcategory	Examples
Identity theft	Identity creation	Trojan horse (such as seeking to gain access to the identity's network)
	Profile cloning	Phishing and scamming
Identity forgery	Disruptive attacks	Trolling (such as misinformation and annoying messages)
	Sockpuppetry	Spreading propaganda, phishing, scamming, and circumventing bans
	Sybil attacks	Increasing reputation, skewing votes, and reviewing scores

IDENTITY-DECEPTION DETECTION METHODS

Several methods have been proposed to detect identity-deception attacks. Some have explicitly aimed at targeting a particular identity-deception category, while others need to be generalized.

This paper classifies methods based on the major approach or theory that guides the detection method.

Immediate-Comparison Methods

Methods that fall under the immediate-comparison category aim to primarily identify cases of profile cloning, sockpuppetry, and Sybil attacks. The general approach compares data directly from a user's account to an existing set of users. It aims to identify cases in which duplicate accounts exist in the system or in which mismatched information exists about a user. The approach can further be divided based on the types of data utilized (verbal and/or non-verbal).

Profile-based methods

Profile-based methods aim to cross-validate account profiles with an existing set of users. The features can be verbal (text) or non-verbal (such as duration between actions). A tested approach using profile-based features identified duplicate accounts in a criminal database.[7] Duplicate records in the data were suspected to be the result of individuals who may have reported false names under separate arrests. The technique traversed through the list, and each record's profile was compared to the rest of the dataset to identify cases based on a profile score and a particular threshold. The technique was highly effective even in the presence of missing data. However, the original method's time complexity was high (asymptomatic time complexity of $O(N)$ for checking a new user against the total user set). The overhead of such a technique could be larger if profile scores for users have to be updated when an account change occurs (such as employment information updated).

A similar approach was used in the social-networking site LinkedIn.[3] The technique utilized profile-based features to identify cases of manual or automatic profile cloning through bots. The method was proposed as a comparison tool meant to be utilized by a user. It analyzed a user's profile and, using LinkedIn's search, attempted to find profiles that may have cloned part of the original. If it were to be applied for all accounts based on real-time changes on accounts, the method would offer high precision at the expense of computational overheads. The estimated time complexity of the method requires an account to be tested against the rest of the dataset ($O(N)$).

Profile-based features can also be used to cross-reference the validity of an account on a social network, A, using information found in a set of anonymized social networks, AG_i.[8] The trustworthiness of features for an account can be evaluated based on anonymized social networks that also contain the account under investigation. The method demonstrated a high accuracy for various network structures; however, the computational overhead of a part of the proposed algorithm had a worst case of $O(|N|^2)$ in respect to the nodes in A.

Verbal-only methods

An alternative approach compares a particular account to the rest of the dataset based on verbal communication. This approach allows for identification of not only cases regarding profile cloning but also cases of sockpuppetry and Sybil attacks.

The approach has been used on Wikipedia to identify sockpuppet accounts.[5] The authors built a natural-language-processing algorithm that analyzed lexical features for a set of Wikipedia accounts to detect likely sockpuppets. The accuracy of the method is considerably high; however, the computational overhead is also high. Comparing a user to all other accounts will result in a time complexity of $O(N)$. But, unlike profile-based approaches, an algorithm will have to traverse for each user, N_i, through all account content, R_i (revisions on article pages in the case of Wikipedia). Effectively, the operation translates to a comparison of all revisions made by users. If no index is used, an algorithm will result in a complexity of $O(N * R)$, while if a binary search tree is used for the revision dataset, the complexity can be reduced to $O(N \log R)$. Platforms such as Twitter have considerably higher proportions of user-content pairs than Wikipedia, rendering the real-time application of this method more impractical.

Baseline Methods

Methods under this category relate to the expectancy-violations theory. The theory looks at how individuals react to unexpected behavior (such as evidence of deception unintentionally leaked by a deceiver). This is translated to a computer deception detector—often using machine-learning algorithms—that is looking for violations of expected behavior, which can often reveal deception. In particular contexts, one can identify a "universal" expected legitimate user behavior, which serves as a baseline. Deviations from this baseline are considered deceptive cues and indicative of identity deception. Contrary to immediate-comparison approaches, baseline approaches require a previously established supervised learning model that is often the result of a particular machine-learning algorithm. The quality of the model and that of the training data will influence the effectiveness of the baseline and the prediction accuracy of a model. These models are more effective at resolving most of the identity-deception cases such as scamming and sockpuppetry. Approaches under this category are further divided into verbal, non-verbal, and hybrid.

Verbal methods

Verbal approaches utilize the content delivered by an individual and look for deceptive cues that are known to be associated with deceptive behavior. Often, these relate to lexical, prosodic, or even sentiment features of speech, whether written or audible.[9] These methods have been extensively applied to spam detection and website scamming attacks.[10] In social media, the approach restricts the attention of the detection algorithm to one account and focuses on verbal data produced from the account. These methods can be extremely effective, as well as computationally efficient ($O(C)$ where C is the content produced by a particular user) since they examine only content produced by one individual. However, their effectiveness is also dependent on the nature of the social-media platform. For example, sentiment analysis in short text is less accurate,[11] so the effectiveness of the method may be limited on Twitter.

The method has been found to be effective in the field of forensics in distinguishing adults posing as children on social networks.[12] Training data establish a baseline for age and gender by utilizing a combination of natural language processing and stylistic language fingerprinting. The approach is highly effective and efficient.

Non-verbal methods

Non-verbal approaches work in a similar manner. They include metadata, summarized actions of an individual, and even metrics that represent user actions or behavior within a particular platform's cyberspace. They have been found to be effective and computationally efficient.[13] In the event that some of the summarized data have already been pre-calculated, the methods can be substantially more efficient than verbal natural-language-processing approaches.

Hybrid methods

Recently, a hybrid approach in which verbal and non-verbal data are combined under one supervised learning model has been proposed, although results have not yet been reported.[14] It is likely that the method could potentially be more effective at detecting identity deception at the expense of efficiency (due to the need for more data processing).

Hybrid Approaches

Recent techniques have attempted to combine the two broader categories to detect identity deception. This is often achieved on the basis of generating a baseline model; however, it often also involves immediate comparisons to existing accounts or other data on the platform. An example of this can be seen through the use of social-network analysis. When a particular user u_i needs to be tested for identity deception, a graph G is built for all users V and connections E between them. Connections can be based on friendships but can also consist of any connection deemed important for an online community (such as followers on Twitter and common articles edited on

Wikipedia). The user's profile in G is compared using various social-network metrics (such as closeness centrality) in respect to all users in V. Then, using a baseline, a decision is rendered on whether the "profile" of u_i matches that of known identity-deception accounts. The decision can be made using supervised learning algorithms or based on an arbitrary metric and a particular threshold.

A recent study has identified, with high precision, Sybil attacks using social-network data.[6] The approach utilized a graph to determine the probable clusters of Sybil and non-Sybil accounts. The time complexity of the approach was $O(n \log n)$. The computational benefits of this approach are based on the fact that it examined the degree of a vertex as opposed to more computationally intensive graph metrics. A similar study proposed a graph approach using a game-theoretic model to explain deception.[15] The study demonstrated the rich potential of utilizing social-network data for deception detection. Implementing similar techniques in real time holds great potential but also poses substantial computational challenges.

REAL-TIME POTENTIAL DETECTION METHODS

This paper posits that, to utilize these methods in real time, one has to consider three primary factors that influence their efficiency: the number of users on a platform, technology infrastructure, and data velocity. Figure 1 depicts a representation of a hypothetical real-time implementation of an identity-deception detection system on social media.

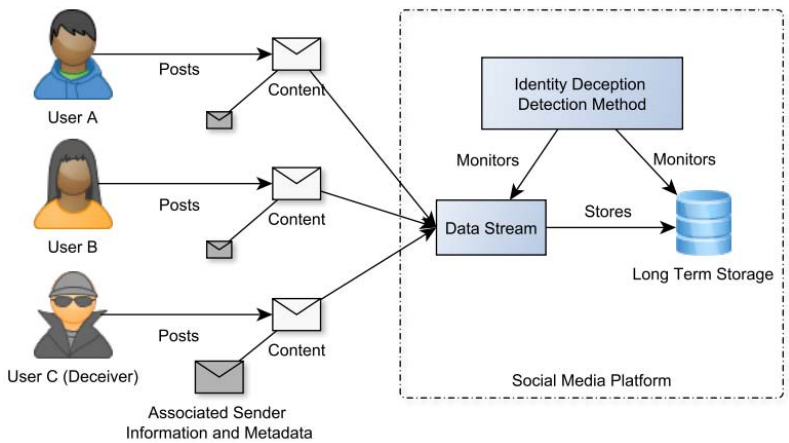

Figure 1. Depiction of an identity-deception detection system on a social-media platform intended to be used for real-time monitoring.

Social-media platforms vary substantially in respect to their user volumes. This will also have an impact on data volume that has to be analyzed by a particular identity-deception detection algorithm. When this needs to occur in real time, the size of the user population may be restrictive for some methods unless optimizations are considered. Additionally, the computational overhead can vary between the methods presented. The overhead may also be influenced by the existing infrastructure. For example, real-time applications have been said to require an active stream that provides an algorithm with the latest changes, but there is also a need for seamless access to older data.[16] If a requirement is not met, some of the methods cannot feasibly be implemented for real-time detection. Finally, when discussing real-time application, data velocity is perhaps the most important factor. For the purposes of this paper, data velocity is defined as the intensity of change in the data. Although different measurements may be used depending on the social-media platform, data velocity describes the flow of data over time. For example, in the context of Twitter, this may be translated as the number of tweets per second. Data velocity will influence the number of times a detection method needs to be executed over a time interval.

Baseline methods are the most efficient. They are independent of the number of users on a platform and require limited support by infrastructure. As such, they can afford to be used in platforms with higher data velocities. Immediate-comparison approaches, on the other hand, can be heavily influenced by the total number of users. As such, when data velocity is high, they will suffer from redundancies that will make real-time application improbable for large social-media platforms. Finally, hybrid approaches are by far the most expensive methods for identity-deception detection. As the total number of users rises, so does the need for substantially more infrastructure to support data processing. For example, social-network analyses are notorious for the memory requirements as graph sizes grow. Each user tested is required to be added to the existing graph, and many metrics will have to be recalculated for all vertices in a graph. Few metrics are not bound by the need for calculating the same metric for all vertices.

OPTIMIZATIONS FOR REAL-TIME OPERATION

This paper identifies several optimizations that can help overcome some of the aforementioned limitations while attempting to preserve most of the effectiveness. The estimated impact on effectiveness is reported based on the literature. The optimizations are meant to be representative, but not exhaustive.

Window Optimization

This optimization aims to apply restrictions on the visibility of data, which effectively reduces the amount of data and the computational overhead while attempting to limit the loss of information—which may cause a reduction in an algorithm's detection accuracy. These methods can impact efficacy positively or negatively depending on the context and the bounds of a system. In the context of real-time detection, it can be further divided based on its application on an active stream of data or complete dataset.

Applying window on stream

While many real-time applications rely on a stream to monitor changes, some attempt to store the most recent set of changes based on a window.[16] This helps reduce the need to query old data directly from a database and provides data available for immediate analysis. The size of this window can be determined based on time (such as changes during the past day) or based on data (such as most recent 1,000 changes). The optimization can benefit baseline approaches, especially in contexts in which recent behavior can be more indicative of a deceptive account than the complete history of an account. However, the optimization is not applicable to methods that rely on account history, as well as immediate-comparison and hybrid approaches.

Applying window on dataset

Methods that rely on accessing past records of data (aside from the active stream) can also be optimized based on windows. Like with the previous optimization focused on the stream, windows can be applied in relation to time or data. The impact of this optimization can radically reduce time complexity for algorithms. For example, a window has been used for detecting duplicate records on a criminal database that reduced the time complexity from $O(N)$ to $O(\log N)$ when testing for a particular user.[7] This came at no cost to efficacy. Similar windows can benefit social-network analysis. For example, localized graph statistics can be calculated based on a restricted view of the overall network based on an n-step reach from the particular user to be tested. This reduces the amount of vertices in a graph and the computational overhead. Finally, the method may yield higher efficacy in baseline approaches, because it could reduce the error caused by the assumption that a community exhibits uniform behavior. As users change behaviors over time, establishing a baseline on recent data may be a more effective strategy.

Adaptive window optimization

Window optimization can be taken a step further. This is often overlooked by literature since the aim is to demonstrate primarily effectiveness of an approach rather than efficiency for real-time implementation. However, social media do not have a static number of users nor a static data velocity. For example, catastrophes can often result in users becoming more active on social-media platforms such as Twitter. Bursts of activity can potentially impact an otherwise functioning real-time deception-detection system. If windows are applied as an optimization technique, having them automatically adjust their size based on data velocity will likely render them more efficient. The tradeoff of accuracy (or inaccuracy) due to limited data can further be remedied by using an adjustable penalty on the probability that an account is deceptive.

Monitoring only Non-Cleared Accounts

This optimization aims to reduce computational overhead without impacting efficiency. It is ideal for baseline methods and can be applied on the stream of data. Actions that come from accounts that have already been cleared as non-deceptive accounts do not have to be monitored. This is likely to occur after an account has been monitored for a said time. For example, a study that used baseline behavior determined that deceptive and legitimate accounts' non-verbal data tend to increasingly deviate from one another with the age of accounts.[13] As such, administrators may decide that an account could be considered cleared after N days of repeatedly "clean" activity. Another study has also reported no cost to accuracy by establishing a trusted set of accounts (or trusted features of accounts).[8] However, a more accurate restriction in most social media would be to clear an account on the basis of actions made by a user, rather than of time. This is because an account can often be old but have limited activity.

Probabilistic Sampling

The aim of sampling is to reduce computational overhead; however, it can come at the expense of effectiveness. It can be applied to immediate-comparison, baseline, and hybrid methods. The implementation can be established on the stream of data and the dataset. The method is similar to those found on airport security random checks or tax-fraud detection. Since real-time deception detection iterates upon the actions made by a user, the likelihood that a deceptive account will be selected over time increases as a user's actions increase. As such, in a uniform random sampling strategy, a balance between the probability threshold and the frequency can balance the probability of eventual detection. Use of probabilistic sampling using game theory can also be a more effective alternative to uniform sampling strategies. Bayesian Stackelberg games have been shown to be effective at preventing an adversary from guessing a random strategy, as well as at providing defenders with the ability to add weights to different types of likely adversaries based on the severity of damage they can inflict (such as vandalism versus sockpuppetry attacks).[17] Furthermore, the efficiency of the optimization can be affected by the size of the user population. It can be adapted to adjust to data velocity or remain independent, although the latter may impact detection accuracy.

Landmark Optimizations

Several approaches that involved social-network analyses[15,18] can also be optimized through improvements to navigational operations in graphs. Often, proximity of an account to other accounts on a graph can be indicative of identity deception. Distance calculation in large graphs found in social-media platforms can be expensive. Instead, if landmark nodes can be selected ahead of time (preprocessing), computational overhead can drastically decrease.[19] The expense of this optimization is the reduced accuracy of a path calculation; however, it can often be negligible depending on the method selected (some also allow for exact estimation). The optimization can be beneficial as the user population grows, since it mitigates the impact of data velocity. It also reduces computational overhead for the infrastructure supporting the detection system.

BALANCING EFFICIENCY AND EFFECTIVENESS

Attaining an ideal balance between efficiency and effectiveness is a considerable challenge that pertains to the application of these optimizations. At a low level, all these methods aim to reduce computational complexity by reducing the data volume that is expected to be parsed by a detection method, which in turn reduces infrastructure requirements. In practical terms, the processing capacity of the detection system needs to be equal or larger to the growth rate of accounts. In social-media applications, the user population often exhibits a growth that follows an exponential trend before transitioning into a long logarithmic tail (for example, Wikipedia's user population growth). Since the underlying system (user population) is finite, the trend over time is often similar to that of probing rates seen by internet worms (for example, Slammer). As such, detection-system optimizations are most useful early on the lifecycle during the linear or exponential growth of a population in which the system will be under the most stress. The loss in detection rates (effectiveness) can be compensated for by either adjusting parameters for these methods or combining them. Some of the tradeoffs and benefits of these optimizations are described below.

Window optimizations reduce computational requirements at the cost of lower effectiveness due to data-volume reduction. However, accuracy may not be affected in the event that data quality is high (such as when there is no missing data) and there is low variance in the data.—in other words, when the difference between attacker and legitimate user is "distinct" in the data. This will also depend largely on the context and how user actions are recorded and compared. For example, we would need larger data windows to determine differences in writing style compared to measuring a user's average time between actions. This is also relevant for cases of weak models (high bias) that do not benefit from the addition of more data.

Clearing accounts after passing a certain trust threshold can also benefit the system from repeatedly evaluating the same users. However, in this approach, false negative accuracy for detection methods becomes particularly important (eliminating doubt on cleared accounts). This reversal in perspective may also require changes in the methods since the aim is not to detect but rather to establish trust. Computationally, the optimization holds promise even if the influx rate of new users surpasses the "clearing" rate. By reducing the threshold for false negatives, the detection system will require fewer resources at the cost of allowing some adversaries to pass undetected.

A similar challenge can also be observed with probabilistic sampling. Random sampling rates have a linear relationship with detection rates in respect to the user population. As such, the computational gains and loss of accuracy are directly comparable. However, the relationship between efficiency and effectiveness becomes more promising when sampling is informed on the basis of indicators (for example, if an account registers in the middle of the night, it is more likely to be a deceiver). As such, biased sampling will likely retain more of the unsampled effectiveness while exhibiting an equal reduction in computational complexity compared to random sampling.

Finally, landmark optimizations in networks hold the potential for reducing processing requirements for graph statistics by building shortcuts for many of the calculations. These often have parameters that define how aggressive algorithms should approximate measurements (such as shortest distance). The relationship between measurement error and landmark parameter size will determine resource requirements and accuracy. In some algorithms, this relationship follows a near-linear trend,[20] and so establishing the loss in effectiveness can be more easily estimated. This is also dependent on how important network statistics are in the original detection method.

CHALLENGES AND OPPORTUNITIES

Identity-deception detection approaches are summarized in Table 2 based on the main purpose and real-time factors that can affect them along with applicable optimizations. Real-time applications for these methods can vary. Most optimizations fit almost all approaches proposed in this paper. The impact on detection accuracy by these optimizations is mainly determined by how aggressive their said parameters will be. Further challenges described in this paper relate to the use of baselines in real-time detection methods and identity management.

Interpersonal deception theory posits that a deceiver's behavior changes over time and adapts based on a victim's responses and environmental factors.[21] These changes are made by a deceiver to ensure deception success. Current deception-detection methods do not look for these temporal adaptations in an account's profile (verbal and non-verbal). Instead, they focus on an account's complete history. However, since accounts are tracked constantly in real-time monitoring, these changes in behavior can be captured and incorporated into algorithmic models. These moves and counter-moves can be used to estimate an attacker's behavior even in active defense scenarios.

Table 2. Identity-deception detection approaches and proposed optimizations.

Identity-Deception Detection Method	Ideal for	Factor(s) with Largest Impact	Optimizations
Immediate-comparison	Profile cloning Sockpuppetry Sybil attacks	User population Data velocity	Window optimization on stream Window optimization on dataset Adaptive windows Probabilistic sampling
Baseline	Identity creation Disruptive attacks Sockpuppetry	Data velocity	Window optimization on stream Window optimization on dataset Adaptive windows Probabilistic sampling Monitoring only non-cleared users
Hybrid	Profile cloning Sybil attacks Sockpuppetry Disruptive attacks	User population Data infrastructure Data velocity	Window optimization on stream Window optimization on dataset Adaptive windows Probabilistic sampling Landmark optimizations

Furthermore, baseline approaches often track behaviors in binary form (deceptive versus non-deceptive account) and assume that all individuals need to behave in accordance to an established baseline. An attacker may be able to trick a detection system by acting legitimately until such time when an attack takes place. Looking for leakage cues may be a more effective and computationally efficient method, but current machine-learning algorithms need large amounts of training data from multiple users and as such are problematic in their application for a single user. Alternative cognitive or game-theoretic models may need to be developed that more closely reflect the internal state of individuals and look for subtle deviations from a predicted legitimate user behavioral path.

Subsequent identity-management approaches that share cross-platform data will need to be developed. Cross-referencing across datasets has been proven to be effective in past studies and can help act as an initial security measure for newly created accounts. This is especially true for cases of identity creation. However, the approach may be limited for cases of identity forgery in

which pseudonymous accounts are allowed by a platform. In such cases, access-control protocols (such as gradually granting privileges to a user) based on real-time trust-management systems may prove to be more effective. An attempt by a user to access a feature beyond the limits set by a system may be an early indicator of the user being an attacker.

Finally, identity-deception prevention can also decrease computational overhead for algorithms. Currently, attempts towards "universal" login credentials through social-media logins and other services such as OpenID aim to accommodate a user demand for a reduction in login credentials. However, these logins could potentially offer summarized identity-deception detection information derived from various websites that they have been used in. Such standardization of reliable identity data (such as times during which or locations from which an individual usually operates) will require extensive studies and collaborations across scientific fields. If successful, a new account created on Facebook for use in other websites will have to demonstrate its legitimacy through its activity pattern on Facebook (or whichever other social login is used).

CONCLUSION

Over the course of a decade, progress has been made in not only understanding how deceivers operate online in social media but also how their behaviors can be tracked and identified. However, successful approaches and algorithm lack plans for implementation in realistic large-scale social-media platforms. This paper describes a classification of existing deception-detection approaches and proposes factors and optimizations that need to be considered for a real-time implementation of these approaches in social media. These recommendations call for a re-examining of detection approaches and their potential use in social media. Given the rapid growth of social media and data velocity, it is likely that technology will not save these detection approaches, but rather will offer innovative new ways to make them practical for real-time applications.

REFERENCES

1. M. Tsikerdekis and S. Zeadally, "Online Deception in Social Media," *Communications of the ACM*, vol. 57, no. 9, 2014, pp. 72–80.
2. M. Tsikerdekis and S. Zeadally, "Detecting and Preventing Online Identity Deception in Social Networking Services," *IEEE Internet Computing*, vol. 19, no. 3, 2015, pp. 41–49.
3. G. Kontaxis et al., "Detecting social network profile cloning," *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2011, pp. 295–300.
4. E. Bertino and K. Takahashi, *Identity Management: Concepts, Technologies, and Systems (Information Security & Privacy)*, Artech House Publishers, 2011.
5. T. Solorio, R. Hasan, and M. Mizan, "A Case Study of Sockpuppet Detection in Wikipedia," *Proceedings of the Workshop on Language Analysis in Social Media*, 2013, pp. 59–68.
6. Q. Cao et al., "Aiding the detection of fake accounts in large scale social online services," *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, p. 15.
7. G.A. Wang et al., "Automatically detecting criminal identity deception: an adaptive detection algorithm," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 36, no. 5, 2006, pp. 988–999.
8. C. Dai et al., "Privacy-Preserving Assessment of Social Network Data Trustworthiness," *International Journal of Cooperative Information Systems*, vol. 23, no. 2, 2014.
9. J. Hirschberg et al., "Distinguishing deceptive from non-deceptive speech," *Interspeech, Proceedings of Eurospeech'05*, 2005, pp. 1833–1836.
10. A. Ntoulas et al., "Detecting Spam Web Pages Through Content Analysis," *Proceedings of the 15th International Conference on World Wide Web*, 2006, pp. 83–92.

11. M. Thelwall et al., "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, 2010, pp. 2544–2558.
12. A. Rashid et al., "Who Am I? Analyzing Digital Personas in Cybercrime Investigations," *Computer*, vol. 46, no. 4, 2013, pp. 54–61.
13. M. Tsikerdekis and S. Zeadally, "Multiple account identity deception detection in social media using nonverbal behavior," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 8, 2014, pp. 1311–1321.
14. A.M. Kuruvilla and S. Varghese, "A detection system to counter identity deception in social media applications," *International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2015, pp. 1–5.
15. A.C. Squicciarini and C. Griffin, "An Informed Model of Personal Information Release in Social Networking Sites," *International Conference on Privacy, Security, Risk and Trust (PASSAT)*, 2012, pp. 636–645.
16. M. Stonebraker, U. Cetintemel, and S. Zdonik, "The 8 Requirements of Real-time Stream Processing," *SIGMOD Rec.*, vol. 34, no. 4, December 2005, pp. 42–47.
17. J. Pita et al., "Using Game Theory for Los Angeles Airport Security," *AI MAGAZINE*, vol. 30, no. 1, 2009, pp. 43–57.
18. C. Peng et al., "Predicting Information Diffusion Initiated from Multiple Sources in Online Social Networks," *Sixth International Symposium on Computational Intelligence and Design (ISCID)*, 2013, pp. 96–99.
19. K. Tretyakov et al., "Fast Fully Dynamic Landmark-based Estimation of Shortest Path Distances in Very Large Graphs," *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (CIKM), 2011, pp. 1785–1794.
20. M. Potamias et al., "Fast Shortest Path Distance Estimation in Large Networks," *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009, pp. 867–876.
21. D.B. Buller and J.K. Burgoon, "Interpersonal Deception Theory," *Communication Theory*, vol. 6, no. 3, August 1996, pp. 203–242.

ABOUT THE AUTHOR

Michail Tsikerdekis is an assistant professor in the Computer Science Department at Western Washington University. His research interests include deception, data mining, cybersecurity, and social computing. Tsikerdekis has a PhD in informatics from Masaryk University. Contact him at Michael.Tsikerdekis@wwu.edu.

Efficient Cloud Provisioning for Video Transcoding: Review, Open Challenges, and Future Opportunities

Maria G. Koziri
University of Thessaly

Panos K. Papadopoulos
University of Thessaly

Nikos Tziritas
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

Thanasis Loukopoulos
University of Thessaly

Samee U. Khan
North Dakota State University

Albert Y. Zomaya
University of Sydney

Video transcoding is the process of encoding an initial video sequence into multiple sequences of different bitrates, resolutions, and video standards, so that it can be viewed on devices of various capabilities and with various network access characteristics. Because video coding is a computationally expensive process and the amount of video in social-media networks drastically increases every year, large media providers' demand for transcoding cloud services will continue rising. This article surveys the state of the art of related cloud services. It also summarizes research on video transcoding and provides indicative results for a transcoding scenario of interest related to Facebook. Finally, it illustrates open challenges in the field and outlines paths for future research.

As reported by Cisco in its mobile-data-traffic forecast for the years 2015 to 2020, mobile Internet traffic increased by 74% during 2015, with 55% of it due to video.[1] Given the popularity of social media, the affordable prices of smart devices with high-resolution cameras, and the ever-

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

lasting user need for social interaction and entertainment, video traffic on the Internet, particularly the mobile one, will continue rising. To further complicate things, large media providers have to satisfy clients viewing video from a plethora of different devices with various players and capabilities that reside behind network connections of various speeds. This necessitates video transcoding, which is the process of encoding an initial video sequence into one or more sequences of different resolution levels, bitrates, quality, and perhaps coding standards—e.g., from H.264/AVC to HEVC. (AVC stands for Advanced Video Coding; HEVC stands for High Efficiency Video Coding.)

In its simplest form, transcoding can be done by decoding the original video and re-encoding it using the desired parameters. Since video encoding is a computationally intensive task, and given that the amount of video that must be transcoded before delivery can exceed the infrastructure capacity of most media providers, using cloud resources seems to be the only scalable solution. Figure 1 shows schematically a generic cloud-transcoding scheme.

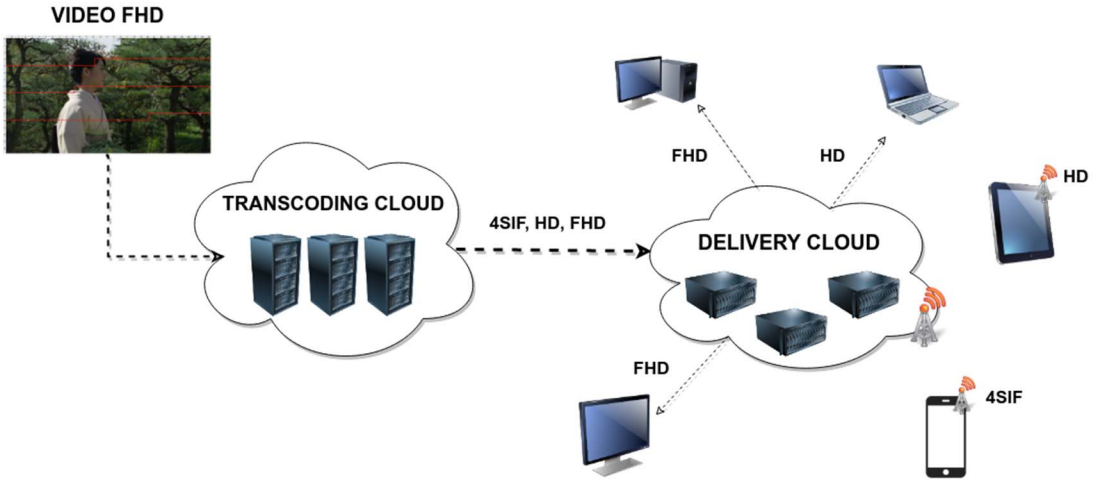

Figure 1. A generic cloud-transcoding architecture. The initial FHD (Full High Definition) video sequence is sent to the transcoding cloud, which produces output sequences in three different resolutions. These sequences are sent for delivery to various devices. SIF stands for Source Input Format.

TRANSCODING AS A SERVICE

A straightforward approach to use cloud resources for transcoding is the IaaS (infrastructure as a service) model, whereby the user should estimate the computational resources required for the tasks; select a codec to use; create, schedule, and monitor job batches; and manage the output streams. This solution entails a non-negligible overhead in specialized personnel costs.

This is why recent years have witnessed a growth in the number of companies offering PaaS (platform as a service) or SaaS (software as a service) solutions, effectively implementing transcoding as a service (TaaS). TaaS aims to reduce the complexity, from a client's point of view, of

- defining transcoding tasks,
- executing them, and
- managing their output.

Concerning defining transcoding tasks, the client is offered predefined encoding modes either optimally tailored for specific devices or tailored for particular resolutions. Such predefined modes can be of huge help to small-scale clients who can't afford the complexity of defining optimal encoding parameters—e.g., bitrate or QP (Quantization Parameter)—for their targeted resolutions. Large media providers can still use their home settings, ignoring the predefined modes.

As far as executing transcoding tasks is concerned, the client is neither required to know the details of the codec used nor required to schedule its tasks. Finally, concerning managing the transcoding tasks' output, TaaS providers can forward videos to a content delivery network after packaging them so that they are ready for streaming—e.g., in MPEG-DASH (Dynamic Adaptive Streaming over HTTP) or HLS (HTTP Live Streaming). Most providers also offer watermarking and DRM (digital rights management) capabilities, as well as options for adding advertisements.

A distinguishing factor among TaaS solutions is whether they offer file or live video transcoding. Another distinguishing factor is the pricing schemes they use. In the case of file-transcoding solutions, providers usually charge for either the output video duration or the output video size, while in live video transcoding, a consumed-bandwidth or per-channel charge is used.

Table 1 summarizes some basic characteristics of related services and gives estimated charging costs for two scenarios: one involving live transcoding and another stemming from VoD (video on demand). The table is not intended to provide a selection list; it doesn't include all the possible providers and solutions, while pricing isn't the single decision criterion. Furthermore, prices usually don't scale linearly to workload size and might also be negotiable.

Table 1. Some cloud-transcoding services.

Company[1]	Codecs	Pricing	Cost for file-encoding scenario[2] ($)	Cost for live-stream-encoding scenario[3] ($)
Brightcove Zencoder	H.264/AVC, VP8, VP9, Theora, HEVC	Output duration, live channel duration	360	N/A
Encoding.com PublicCloud	H.264/AVC, VP8, Theora, HEVC	Output size, per-channel flat price	1,250	899
Amazon Elastic Transcoder	H.264/AVC, VP8, VP9	Output duration	180	N/A
Telestream Cloud	H.264/AVC, VP8, VP9, Theora, HEVC	Output duration	239	N/A
Wowza Streaming Cloud	H.264/AVC, HEVC	Processing time, output size, per-channel rate, no. of streams, stream duration	N/A	856
Microsoft Azure Media Services	H.264/AVC, Theora, HEVC	Output size, live channel duration	180	801
Bitmovin	H.264/AVC, VP9, HEVC, AV1	Output size	156	468

1. The information is based on companies' websites. In the price-charging scenarios, the most economic package was selected. Prices are for coding; other charges such as storage, transmission, etc. might apply.
2. The scenario consisted of transcoding 100 hour-long videos into 100 FHD (Full High Definition) output sequences. Each output sequence was estimated to be 25 Gbytes.
3. The scenario consisted of setting a live stream. The input stream was assumed to be transcoded into a single output stream of FHD quality. The total channel duration considered was 300 hours monthly.

THE CASE OF SCALABLE VIDEO CODING

A problem with transcoding is that a single video sequence leads to multiple output files. Scalable Video Coding (SVC) aims to solve this problem by coding in one bitstream multiple fidelity points of the original video signal. The fidelity points can be the temporal resolution, spatial resolution, or quality (the signal-to-noise ratio—SNR).

Thus, the bitstream comprises layers—i.e., subsets of the bitstream that represent the additional information needed to be able to decode the signal at an increased quality. Layers form a hierarchy whereby a specific layer needs all the lower ones it refers to, in order to be decoded. Therefore, each layer, in combination with the layers it depends on, forms a representation of the video signal in a specific spatial–time resolution and quality.

For a detailed overview of SVC extensions in H.264/AVC, see "Overview of Scalable Video Coding Extension of H.264/AVC Standard."[2] SVC has seen some successful deployment in the industry—e.g., Vidyo's conferencing service. Nevertheless, it hasn't replaced transcoding owing to its lack of popularity and interoperability difficulties. Additionally, transcoding is necessary when changing the video standard.

RESEARCH ON CLOUD TRANSCODING

Most TaaS providers don't give details concerning algorithms and system architecture design. So, we focus here on surveying research from academia.

Efficient Transcoders

A significant amount of work exists on how to efficiently implement transcoders. When the requirement is to produce a video sequence in the same format but presumably at a lower bitrate (also called *transrating*), a straightforward yet inefficient approach is to use an open-loop method whereby the decoded sequence is re-encoded using larger QP values. Closed-loop methods involving error compensation have also been proposed. A survey on transrating can be found in "Video Transcoding: An Overview of Various Techniques and Research Issues."[3] Of particular interest is the case in which a sequence must be transcoded using different video standards.

In "Fast Video Transcoding from HEVC to VP9," the authors propose a scheme for transcoding HEVC to VP9.[4] By exploiting information from the HEVC decoding process (mainly, the Intra and Inter prediction modes and the reference frames), they pruned decisions of a VP9 encoder, improving the time up to 60%.

In "Fast H.264 to HEVC Transcoder Based on Post-order Traversal of Quadtree Structure," an H.264/AVC-to-HEVC transcoder is presented.[5] The proposed scheme achieves a speedup of 7.89 times compared to full re-encoding, while the coding efficiency loss is 3.28% BD-Rate. (BD stands for Bjøntegaard's Delta.) This speedup is achieved by two modifications in the HEVC encoder. First, a fast mode decision framework based on a post-order traversal of the CTU (Coding Tree Unit) quadtree is proposed. This framework, in combination with information from the H.264/AVC sequence and a modified RD (rate distortion) cost prediction model, achieves early termination of the mode decision process. Second, a new fast motion estimation algorithm is implemented that selects the best candidate from a list of previously encoded H.264/AVC and HEVC motion vectors.

Another H.264/AVC-to-HEVC transcoder is presented in "Fast Quadtree Level Decision Algorithm for H.264/HEVC Transcoder."[6] Once again, the target is to accelerate the mode decision process in HEVC by using information available in the H.264/AVC sequence. This is done by a Fast Quadtree Level Decision (FQLD) algorithm. FQLD exploits the information gathered at the H.264/AVC decoder to decide on CU (coding unit) splitting in HEVC using a naive Bayes probabilistic classifier. The results show that a speedup of up to 3.98 times is achievable without significant RD loss.

System Design and Scheduling of Transcoding Jobs in the Cloud

In *Video Processing in the Cloud*, a MapReduce approach for video encoding is described.[7] The method is based on

- splitting an initial sequence into chunks, whereby the chunk size is a multiple of GOP (Group of Pictures) size;
- processing each chunk separately; and
- merging them in a final step to produce the final compressed video sequence.

Instead of having a fixed number of GOPs per chunk, the authors of "Dependency-Aware Distributed Video Transcoding in the Cloud" propose to adapt the chunk size.[8] The premise is to avoid breaking dependencies (at chunk boundaries) between GOPs of the same scenery.

In "Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices," a cloud-based transcoding system is presented.[9] The system architecture consists of

- a task manager that accepts user requests and checks whether they can be satisfied by the cached files,
- transcoding servers,
- downloaders that are responsible for locating and fetching a video that isn't already cached, and
- a task dispatcher controlling the rate of downloading and transcoding activities.

A significant part of transcoding tasks is performed during light load periods (e.g., nighttime) as a prefetching strategy.

In "ME-VoLTE: Network Functions for Energy-Efficient Video Transcoding at the Mobile Edge," the authors propose a system architecture that enables video encoding at the edges of a mobile network.[10] They show that energy savings in mobile devices can be achieved by shifting some of the encodings toward edge servers. The core of their proposal is to tune the sending devices to encode videos at a high bitrate, fast, thus saving energy (but consuming larger bandwidth). The videos will then be transcoded at the edges for final delivery.

The focus of "Optimizing the Video Transcoding Workflow in Content Delivery Networks" is to reduce the computational load of transcoding.[11] For each video (or the popular ones), the first chunk is transcoded in an offline manner to reduce buffering. The remaining chunks are transcoded in an online manner using a Markov chain estimator that decides which video parts will be needed once downloading starts.

Partial transcoding is also the focus of "Towards Cost-Efficient Video Transcoding in Media Cloud: Insights Learned from User Viewing Patterns," whereby the aim is to optimize the long-term operational cost of a video delivery service.[12] This cost comes in terms of storage by caching transcoded video segments and in terms of the computational effort due to transcoding uncached segments. An online algorithm is proposed that decides which segments should be cached on the basis of Lyapunov optimization and queuing theory. It is shown that the operational cost can be reduced by 30%.

An admission control algorithm for transcoding requests arriving at a cluster is proposed in "Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing."[13] The algorithm might choose to either accept a transcoding job if servers' working queues are lightly loaded or defer the request, redirecting it to an entertainment server. Deferred requests are rejected when the entertainment server becomes overloaded or are accepted in the next admission slot if the server queue length allows it.

Live video transcoding is considered in "Transcoding Live Adaptive Video Streams at a Massive Scale in the Cloud," focusing on the case in which sources don't have strict high QoS (quality of service) demands.[14] The authors analyze datasets from Twitch concerning livecasting and tackle

the problem of optimizing the user experience. A perfect user experience is achieved by transcoding to the maximum user-supported resolution and bitrate. The authors have developed an ILP (integer linear program) formulation to decide the formats to be transcoded for every stream, given the existing hardware.

Research Summary and Discussion

The aforementioned research targets optimization of the operation of transcoding clouds by

- speeding up the transcoding time of a single task,
- achieving predefined quality levels,
- achieving real-time performance or performance based on a service-level agreement,
- scheduling multiple transcoding tasks optimally, and
- reducing resource and energy consumption.

These goals often conflict with each other; hence, there is a need for further research to develop methods that achieve better tradeoffs.

FUTURE CHALLENGES

A number of future challenges exist, especially as more and more video traffic will concern sequences at 4K resolution. Here, we present some of them.

Efficient Cloud Resource Management

Research on cloud-transcoding resource management is far from being a closed topic, especially as energy consumption and network overhead become increasingly important. An urgent need exists for holistic approaches to system-level design and algorithmic concepts that tackle in a unified way different transcoding scenarios (e.g., standalone files, batches of VoD transcoding tasks, and livecasting), because the relevant market can't be overlooked by TaaS providers. Such unified approaches should also include SVC (wherever applicable). It is also apparent that research exploiting the potential of edge computing in minimizing network and computational overhead will be of paramount importance in an era of 4K and 8K resolutions.

Per-Sequence Transcoding Ladders

In most cases, media providers fix the bitrates and resolutions for transcoding (also called the encoding or transcoding ladder). As advocated by Netflix, one-size-fits-all is not necessarily the best approach.[15] In fact, it was found that movies have different bitrate demands depending on their category. For instance, cartoons could be compressed with reasonable quality using less than half the bitrate of other movies. Research in this area, especially when combined with user-perceived quality metrics, promises to reduce the bitrates of encoded videos without affecting quality.

Adoption of Newer Video Standards

Increased compression efficiency at the same quality is the premise behind any newly developed video standard. HEVC, together with AV1 (AOMedia Video 1) and the planned VVC (Versatile Video Coding), will form the state of the art of video-coding standards in the coming years. The compression performance of the new standards will likely trigger a surge in transcoding demand in the foreseeable future.

The scale as well as the costs involved in the above endeavor could be massive. YouTube experiences a workload of more than 300 hours of uploaded videos per minute. Assume that all the videos uploaded in a year must be transcoded in a new standard, with one output sequence per

input file. By using the pricing scheme of Amazon's Elastic Transcoder, the total financial cost would be between $142 M and $284 M. If more output sequences were needed, the budget could scale in the order of billions. Computational demand will also be huge. Assuming the assigned virtual machines (VMs) achieve real-time performance (24 frames per second), 18,000 VMs working 24/7 for a whole year would need to be allocated.

From a media provider's point of view, research on efficiently using the available cloud resources to maintain QoS in a cost-effective manner will be of primary importance. This will likely entail careful selection of the videos to be transcoded. As the next section illustrates, for a popular social-media network, this can be achieved at a relatively small cost.

A FACEBOOK EXPERIMENT

Here we characterize the impact on Facebook traffic by moving from H.264/AVC to HEVC. We used the x264 and x265 codecs for H.264/AVC and HEVC, respectively, with PSNR-tuned coding settings suggested by "A Large-Scale Video Codec Comparison of x264, x265 and libvpx for Practical VOD Applications."[16] (PSNR stands for peak signal-to-noise ratio.) Experiments were run over four identical servers, each with two six-core Intel Xeon E5-2630 CPUs at 2.3 GHz with 256 Gbytes of memory.

The Dataset

To the best of our knowledge, no suitable dataset existed for the experiment we wished to conduct. So, we created our own. We collected 200 Facebook videos by selecting the top 20 videos from each of the top 10 video publishers as they appeared in the rankings of May 2016 at https://tubularlabs.com. We downloaded each video in the highest-available resolution, which for 168 of the videos was 720p or higher. All videos were in the MP4 format.

For the experiment, we performed transcoding by decoding each video using the x264 codec and re-encoding it in HEVC using x265. Table 2 details the video publishers, the dataset characteristics, and the results of the transcoding experiment.

Table 2. Transcoding a Facebook dataset into HEVC (High Efficiency Video Coding).

Publisher statistics	
Names	Tasty, UNILAD, The LAD Bible, Viechten met Daan, NowThis, Tastemade, FailArmy, CH51, PlayGround Video, Nifty
Categories	Food (2), Entertainment (1), News (4), Style & Beauty (1), Magazine (2)
Total no. of views in May 2016 (millions)	9,304
Dataset statistics	
No. of videos	200 (top 20 per publisher)
Resolutions	23 different resolutions, 720p (82 videos), 1080p (62 videos)
Total no. of views (millions)	3,410
Average no. of views per video (millions)	17.04
Total video size (Mbytes)	1,352.66

Publisher statistics				
Average video size (Mbytes)	6.76			
Total video duration (s)	11,984			
Average video duration (s)	59.92			
Total no. of frames	325,962			
Total network overhead (Tbytes)	24,509.52			
Transcoding results				
	Total video size (Mbytes)	**Total network overhead (Tbytes)**	**Average peak signal-to-noise ratio**	**Aggregate core utilization time (s)**
H264/AVC to HEVC	708.21 (47.64% reduction)	11,435.19 (53.34% reduction)	44.35	14,272,397

Before proceeding to the results, we discuss how well the dataset characterized Facebook traffic, using the following observations:

- The total number of views for the top 10 publishers on May 2016 was 9,304 M. Using the average video size calculated in the dataset (59.92 secs), these views translate into approximately 5 M hours of video views per day.
- Our dataset accounted for 36.6% of this volume.
- As of January 2016, it was reported (at https://techcrunch.com) that an estimated 100 M hours of video were viewed daily on Facebook.[17]

Even if the explosive growth of Facebook means that this value probably became much larger on May 2016, the top 10 publishers still accounted for a considerable ratio (5% with Jan. 2016 statistics), of which the dataset accounted for roughly one-third. Therefore, we can state that the collected dataset captured a sufficient portion of Facebook video traffic.

The Results

The experiment attempted to answer these two questions:

- What would the gains be if instead of H.264/AVC, HEVC was used?
- What are the related computational and financial overheads?

To answer these questions, dataset videos were transcoded into HEVC using their original resolutions. The results indicate that compared to H.264/AVC, the aggregate video size was reduced by 47.64%. Network overhead was calculated assuming that all views were for the maximum resolution and by counting only the video file sizes (the extra cost of network packaging was rather negligible). Table 2 records a reduction of 53.34% compared to the original sequences, while the PSNR remained high.

The runtime results reflect the aggregated execution times of the transcoding tasks when using one thread each. In order to finish the 200 transcodings, the four available 12-core servers should work continuously at maximum core utilization (12 tasks per server) for 3.44 days. The C5 EC2 instance is suggested by Amazon for computing demand tasks. (EC2 stands for Elastic Compute Cloud.) Because the PassMark ratio per core between the Intel Xeon Platinum CPU (c5.large)

and E5-2630 (our servers) is 1.34, it is expected that (all other things being equal) running the tasks on 24 c5.large instances (48 cores total) would require approximately 2.49 days, for an on-demand cost of $137.68. Alternatively, by using a TaaS provider, the cost for transcoding the dataset—e.g., in the case of Amazon's Elastic Transcoder—will roughly be $7.

Bearing in mind that our dataset accounted for roughly 1.8% of one month's total view load (as of Jan. 2016), the results indicate that social-media networks tailored for exchanging short-duration videos will benefit dramatically by moving to a newer standard. These benefits can be materialized by transcoding only a small portion of the most popular videos at low prices, using TaaS providers.

ACKNOWLEDGMENTS

Samee U. Khan's work was supported by the US National Science Foundation (NSF). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. Nikos Tziritas's work was supported by the National Natural Science Foundation of China and PIFI (CAS President's International Fellowship Initiative) International Scholarship under grants 61550110250 and 2017VCT0001, respectively. Panos K. Papadopoulos was supported by a scholarship from IKY (State Scholarships Foundation) funded by the Act "Strengthening Human Resources Research Potential via Doctorate Research" of the Operational Program "Human Resources Development Program, Education and Lifelong Learning," 2014–2020, cofinanced by the European Social Fund (ESF) and the Greek government.

REFERENCES

1. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020*, white paper, Cisco, 3 February 2016; https://www.cisco.com/c/dam/m/en_in/innovation/enterprise/assets/mobile-white-paper-c11-520862.pdf.
2. H. Schwarz, D. Marpe, and T. Wiegand, "Overview of Scalable Video Coding Extension of H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, September 2007, pp. 1103–1120.
3. I. Ahmad et al., "Video Transcoding: An Overview of Various Techniques and Research Issues," *IEEE Transactions on Multimedia*, vol. 7, no. 5, October 2005, pp. 793–804.
4. E. de la Torre, R. Rodriguez-Sanchez, and J.L. Martínez, "Fast Video Transcoding from HEVC to VP9," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 3, August 2015, pp. 336–343.
5. J.F. Franche and S. Coulombe, "Fast H.264 to HEVC Transcoder based on Post-Order Traversal of Quadtree Structure," *Proc. 2015 IEEE International Conference on Image Processing* (ICIP 15), 2015, pp. 477–481.
6. A.J. Díaz-Honrubia et al., "Fast Quadtree Level Decision Algorithm for H.264/HEVC Transcoder," *Proc. 2014 IEEE International Conference on Image Processing* (ICIP 14), 2014, pp. 2497–2501.
7. R.S. Pereira and K. Breitman, *Video Processing in the Cloud.*, Briefs in Computer Science, Springer, 2011.
8. M.R. Zakerinasab and M. Wang, "Dependency-Aware Distributed Video Transcoding in the Cloud," *LCN 2015*, 2015, pp. 245–252.
9. Z. Li et al., "Cloud Transcoder: Bridging the Format and Resolution Gap between Internet Videos and Mobile Devices," *Proc. of ACM Conference on Network and Operating System Support for Digital Audio and Video* (NOSSDAV 12), 2012, pp. 33–38.
10. M.T. Beck et al., "ME-VoLTE: Network Functions for Energy-Efficient Video Transcoding at the Mobile Edge," *Proc. 18th Conference on Innovation in Clouds, Internet and Networks* (ICIN 15), 2015, pp. 38–44.

11. D.K. Krishnappa, M. Zink, and R.K. Sitaraman, "Optimizing the Video Transcoding Workflow in Content Delivery Networks," *Proc. ACM Multimedia Systems Conference 2015* (MMSys 15), 2015, pp. 37–48.
12. G. Gao et al., "Towards Cost-Efficient Video Transcoding in Media Cloud: Insights Learned From User Viewing Patterns," *IEEE Transactions on Multimedia*, vol. 17, no. 8, August 2015, pp. 1286–1296.
13. A. Ashraf et al., "Stream-Based Admission Control and Scheduling for Video Transcoding in Cloud Computing," *Proc. IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing* (CCGrid 13), 2013, pp. 482–489.
14. R.A. Pardo et al., "Transcoding Live Adaptive Video Streams at a Massive Scale in the Cloud," *Proc. Proc. ACM Multimedia Systems Conference 2015* (MMSys 15), 2015, pp. 49–60.
15. A. Aaron et al., "Per Title Encode-Optimization," *Netflix Tech blog*, blog, 14 December 2015; http://techblog.netflix.com/2015/12/per-title-encode-optimization.html.
16. J. De Cock et al., "A Large-Scale Video Codec Comparison of x264, x265 and libvpx for Practical VOD Applications," *Proc. of SPIE, Applications of Digital Image Processing XXXIX*, 2016, pp. 9971–997116.
17. J. Constine, "Facebook Hits 100M Hours of Video Watched a Day, 1B Users On Groups, 80M on Fb Lite," TechCrunch, 27 January 2016; https://techcrunch.com/2016/01/27/facebook-grows.

ABOUT THE AUTHORS

Maria G. Koziri is an assistant professor in the Department of Computer Science at the University of Thessaly (UTH). Her research interests relate to video compression: codec optimization, hardware design, and scalable video coding. Koziri received a PhD in computer science from UTH. Contact her at mkoziri@uth.gr.

Panos K. Papadopoulos is a PhD student in the Department of Computer Science and Biomedical Informatics at the University of Thessaly (UTH). His research interests include video coding and cloud computing. Papadopoulos received his BS from the UTH Department of Computer Science and Biomedical Informatics. Contact him at ppapadopoulos@dib.uth.gr.

Nikos Tziritas is a researcher at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. His research interests include scheduling, load balancing, and replication in content delivery networks as well as energy optimization and resource management in wireless sensor networks and cloud-computing systems. Tziritas received his PhD in computer science from the University of Thessaly. Contact him at nikolaos@siat.ac.cn.

Thanasis Loukopoulos is an assistant professor in the University of Thessaly's Department of Computer Science and Biomedical Informatics. His research interests include green, cluster, grid, and cloud computing. Loukopoulos received a PhD in computer science from the Hong Kong University of Science and Technology. Contact him at luke@dib.uth.gr.

Samee U. Khan is associate professor of electrical and computer engineering at North Dakota State University. His research interests include the optimization, robustness, and security of cloud, cluster, and big data computing; social networks; wired and wireless networks; power systems; and optical networks. Khan received a PhD in computer science from the University of Texas at Arlington. He's a Fellow of the Institution of Engineering and Technology and British Computer Society and a Senior Member of IEEE. Contact him at samee.khan@ndsu.edu.

Albert Y. Zomaya is the Chair Professor of High Performance Computing & Networking at the University of Sydney's School of Information Technologies. His research interests include parallel, distributed, and mobile computing and complex systems. Zomaya received a PhD in control engineering from Sheffield University. He's a Fellow of IEEE, the Institution of Engineering and Technology, and the American Association for the Advancement of Science. He previously was the editor in chief of *IEEE Transactions on Computers*. Contact him at albert.zomaya@sydney.edu.au.

Low-Latency Networking: Architecture, Techniques, and Opportunities

Xutong Zuo,
Tsinghua University

Yong Cui
Tsinghua University

Mowei Wang
Tsinghua University

Tianxiao Xiao
Syracuse University

Xin Wang
Stony Brook University

Editor:
Yong Cui
cuiyong@tsinghua.edu.cn

With the advent of delay-sensitive applications, low-latency networking is attracting research attention from academia, industry, and standards organizations. This article analyzes the causes of latency across network architecture, reviews some state-of-the-art techniques to reduce latency, and presents several opportunities.

The emergence of new applications and operational scenarios places exacting requirements on latency. For example, high user-perceived latency in a cloud game deteriorates players' interactions and degrades the user experience. In industrial operations, control systems depend on low network latency, which is often required to be within several to hundreds of milliseconds to achieve real-time control.[1]

However, the Internet was designed originally to provide best-effort delivery and does not guarantee any quality of service (latency included). In addition, network latency is impacted by many factors such as routing decisions and network traffic. As a result, achieving low network latency is a challenging but critical problem that has attracted great attention. For instance, the former chair of the Internet Engineering Task Force (IETF) discussed design choices of applications that require low latency from a system perspective.[2] Moreover, a broad survey was organized to classify techniques by latency sources, which provided a comprehensive understanding of the root causes of latency.[3]

Unlike previous taxonomies, we analyze latency and techniques by following layers of the network architecture, which naturally integrates with network standards. We attempt to present the issue of latency from an architectural perspective, and in doing so, we hope to facilitate the development of IETF standards.

Both the requirements and the sources of latency vary with different functionalities on different layers. Utilizing services provided by lower layers, upper layers with more functionalities will introduce additional delay. On the application layer, many applications focus on the completion time of transmitting a data block that might consist of several packets. On the transport layer, if

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

reliable and ordered transmission is provided, retransmission delay and head-of-line (HOL) blocking are added on the basis of the round-trip time (RTT).

Lower layers focus on the latency of delivering every packet. Routing on the network layer determines the paths, leading to queuing delay. The link layer is responsible for transferring datagrams between adjacent network nodes, where latency consists of channel access delay due to the shared medium.

Here, we present a brief survey of network latency and approaches to reduce latency, particularly the delays resulting from the protocol design and functionalities. We first summarize factors impacting delay from different layers of the network architecture. Then, we review some state-of-the-art solutions to reduce latency at each layer. Finally, opportunities for future work and concluding remarks are given.

LATENCY ACROSS THE NETWORK ARCHITECTURE

In this section, we analyze the main causes of latency at each layer of the TCP/IP architecture. As Figure 1 shows, each layer involves various functions that trigger latency, such as congestion control on the transport layer and routing on the network layer. To implement functions, related protocols have been designed, and mechanisms in these protocols introduce delays, such as TCP handshake delay or loss recovery delay. Note that latency and delay are used interchangeably in this article.

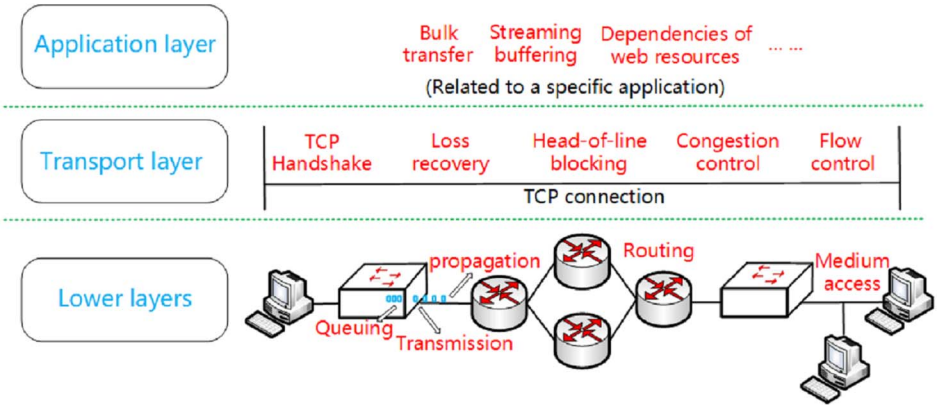

Figure 1. Latency at each layer of the TCP/IP architecture. Different causes of latency and factors affecting latency are shown according to the network architecture.

Latency Specific to Applications

Applications with disparate design goals can have different sources of latency. One of the most important concerns of latency is the block completion time. Only when all units in the block (e.g., chunks of file transfers and frames in video streaming) are transferred to the receiver can the data be used by the application. In the following text, we take two typical applications as examples to introduce other sources of delay.

Interactive live streaming demands low communication latency, which is different from the traditional video-on-demand application. Viewers who comment on broadcasts need immediate feedback; the Real-Time Messaging Protocol (RTMP) is applied to help achieve low latency. The end-to-end delay from broadcasters to viewers adopting RTMP includes upload delay (the delay for transferring a frame from the broadcaster to the server), last-mile delay (the delay for transferring a frame from the server to the viewer), and client-buffering delay (the gap between the frame arrival time and frame playing time).[4] The client buffer aims to prevent playout interruption, but a long buffering delay on the application layer is introduced, which accounts for the largest proportion of the end-to-end latency.

For web services, the page-loading time can be defined as the duration from the receipt of the user request to the display of the whole page. Measurements and analyses have shown that latency plays a defining role concerning page-loading time as well as bandwidth.[5] Factors that contribute to high latency on the application layer include the TLS/SSL handshake (required for secure connections) delay, HTTP redirections (redirecting a client request to a different location), HTTP blocking (the waiting time caused by the maximum number of TCP connections to a server in a browser), and dependencies between web resources (evaluating previous objects and fetching new resources).[6]

Latency of Data Transmission on Lower Layers

The latency on the transport layer is determined by the transport mechanism. Reliable, ordered transmission incurs high latency; we focus on per-connection latency in this case. The per-connection latency comprises the protocol handshake, which creates a connection setup delay that bears on the overall transmission delay, especially for short flows. Moreover, to ensure reliable and ordered delivery of a series of packets, the retrieval and retransmission of a lost segment incur a tremendous delay, which also produces HOL blocking by holding up subsequent segments.

Routing is a core functionality of the network layer. Unlike the other sources of delay presented in this article, the latency of the routing operation generally has been neglected, but routing decisions greatly impact latency. A selected path with more hops might introduce higher packet processing and propagation delay as packets traverse more network devices. Besides, the queuing delay fluctuates significantly with the routes, leading to varying RTTs.

For a shared medium, media access control (MAC) on the link layer addresses the shared-channel-allocation problem. Additional latency arises from the design of MAC protocols. Static channel allocation, such as time division multiple access (TDMA), gives rise to predictable latency due to the fixed assignment of the shared medium. For TDMA, the sender waits for its assigned time slot, rendering the unused time slot idle and thus increasing the waiting time, especially when the channel load is low. As for dynamic allocation, the collision and buffering delay resulting from the contention channel cannot be ignored, especially when the channel load is high. Channel competition makes the transmission delay of a frame erratic, which is not suitable for real-time traffic.

The queuing delay at each device contributes a large part to the end-to-end latency, which is caused by the contention of shared sources, such as output ports and processing units. The buffer is set to hold queued packets, preventing packet loss and handling network burst traffic, which incurs a queuing delay. Many factors affect the queuing delay, including routing decisions, congestion control, etc. For longer links, such as satellite links, propagation delay is nontrivial on the lower layers.

NOVEL TECHNIQUES TO REDUCE LATENCY

To achieve low network latency, we can consider the delay created at each layer of the TCP/IP network architecture. In this section, we discuss some research efforts dedicated to reducing the latency mentioned above.

Reducing Application-Specific Latency

First, we introduce two typical applications whose application layer latency we discussed earlier: web services and interactive live streaming. Then, we present the corresponding methods on the application layer for reducing the latency.

For web services using HTTP, unnecessary handshakes incur latency during connection setup. Many procedures performed on the application layer improve the handshake delay by reducing the number of TCP connections. HTTP 1.1 adopts persistent connections to deal with multiple HTTP requests in one TCP connection.[7] However, the support for concurrent connections brings latency and gives rise to mounting complexity of management.

Fortunately, the multiplexing adopted by HTTP 2.0[8] addresses this problem. Several requests or responses sent to the same receiver are multiplexed while being transferred. Labels are applied to distinguish between different streams and different packets. This eliminates the need for establishing multiple TCP connections and contributes to the reduction of page-loading time. In fact, HTTP 2.0 has received the attention of the IETF and was standardized by the HTTPBIS (Hypertext Transfer Protocol) Working Group.

For live streaming, smooth playout and low latency are conducive to a good user experience. To achieve continuous playout when the condition of the best-effort network varies, buffering on the client side is employed. However, an inevitable delay comes with the buffer. Measurements show that the buffering latency is the largest part of the live streaming delay.[6]

To reduce buffering latency without suffering from stalls, the amount of prebuffered data and the adjustment of the playout rate are worthy of study. For example, adaptive media playout is an effective methodology that aims at playout rate adjustment and has proven highly successful in buffer latency reduction. Specifically, an attempt has been made to leverage historical buffer level variation information and estimate the buffer variation range, after which buffer size is adjusted accordingly.[9] This method reduces the latency under good conditions, using a shallow buffer to guarantee no interruption.

Optimizing the TCP Handshake and Alleviating HOL Blocking

For a single TCP connection, the traditional three-way handshake adds latency to data transmission, especially for short flows. This latency can be mitigated by reducing the number of control interactions in a single TCP connection. TCP Fast Open (TFO) aims to start transmitting data while carrying on the handshake.[10] It is achieved by using a cookie and sending data to the receiver before an acknowledgment arrives. If a TLS handshake is required to provide a secure connection, one to two more RTTs are introduced on top of conventional TCP.

Quick UDP Internet Connections (QUIC)[11] can achieve one RTT handshake for the first-time connection by incorporating the transport and crypto handshake. For subsequent connections, clients send the cached cryptographic cookie, the encrypted payload, and other information to the server, which the server can utilize to authenticate clients and decrypt the payload data. As a result, a zero-RTT handshake is attained. Internet statistics show that QUIC was supported by about 1.0 percent of all websites as of August 2018.[12]

Another delay contributor on the transport layer is HOL blocking, which is caused by lost or out-of-order packets in the sequential packet delivery. Methods that speed up the retransmission of lost packets help relieve HOL blocking. Keeping the sender aware of the packet loss in a timely manner, instead of waiting for the retransmission timeout, can accelerate the retransmission. In fast retransmit,[13] three (or fewer in early retransmit[14]) duplicate ACKs indicate packet loss and trigger the retransmission. An explicit notification of packet loss can also notify the sender to retransmit with dispatch. The cutting-payload mechanism trims the packet, leaves the header transferred to the receiver, and sends negative acknowledgments to inform the sender of the packet loss.[15] By decreasing the time to retransmit the lost packet, the waiting time of subsequent packets is reduced.

Some new protocols also have countermeasures upon HOL blocking. For instance, QUIC deals with this problem by supporting multiple streams in one connection. To be more specific, one QUIC packet consists of several frames of a small number of streams. Thus, a lost packet only causes HOL blocking of the corresponding streams, rather than all streams.

Also, in Multipath TCP, because the packets take different paths with different RTTs, they might arrive at the receiver out of order. To solve this problem, opportunistic retransmission has been proposed, in which the message causing the HOL blocking is re-sent on subflows that have available congestion windows.[16]

Moreover, another method, Slide Together Multipath Scheduler, allows preallocating packets for the fast path and allows packets with a larger sequence number to travel on slow paths.[17] In this

way, packets can arrive at the receiver in order without HOL blocking; more details can be seen in Figure 2.

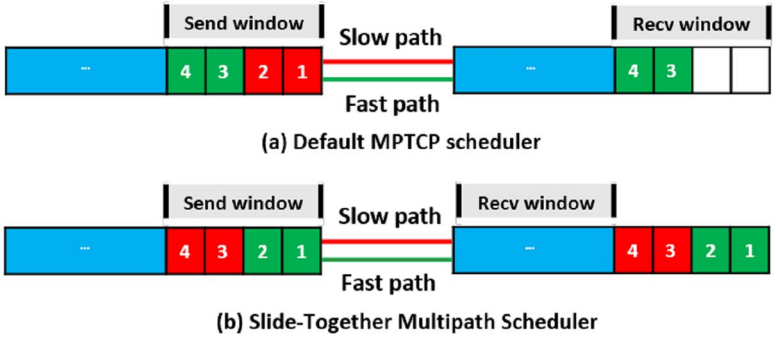

Figure 2. The Slide Together Multipath Scheduler (STMS) prevents HOL (head-of-line) blocking. If the fast path is not available when the sender starts sending data, the default Multipath TCP (MPTCP) scheduler will send packets with small numbers on the slow path, which causes HOL blocking. However, when STMS is adopted, the slow path always sends packets with sequence numbers larger than those of the fast path.

Reducing Latency on the Lower Layers

Routing is a fundamental functionality on the network layer that affects RTT by selecting paths. However, existing routing protocols were designed without considering latency. Methods based on traffic engineering can be used to reduce latency, but they are often limited by inadequate models. Queuing theory is a common way to model network queuing and assist traffic routing. Nonetheless, it might not perform well because in most cases it is a queued network rather than a single queue that is dealt with. A model-free approach utilizes deep reinforcement learning, bypassing the problem of building an accurate mathematical model.[18] Despite the potentially high performance, it might face some challenges; e.g., it is hard to obtain the required real-time reward.

Some companies, such as ViewQwest and Amazon, are also studying latency-based routing to reduce RTT and meet the requirement of delay-sensitive applications. For example, ViewQwest benefits from redundancy by monitoring traffic and probing all available paths to select the best route for low latency.[19]

Frame aggregation is an effective enhancement to improve throughput and reduce medium access delay on the link layer. It reduces the transmission delay and buffering delay by abating the overhead of the frame header and the number of contentions, respectively. However, a tradeoff exists between the time spent in computing aggregates and the time saved owing to aggregating frames. Operation with high efficiency might be complicated and consume much more time.[20] In addition, a larger aggregated frame means a longer waiting time before channel access. Existing aggregation methods mostly aim at improving throughput and transmission efficiency. Reducing latency while optimizing the above two performance metrics is still an open issue. Frame size adaptation considering channel conditions might also be a choice.

Reducing Queuing Delay

Queuing delay is one of the most prominent factors that contribute to high end-to-end latency, especially for datacenters with small internal distances. A priority mechanism can reduce flow completion time for delay-sensitive flows.[21] To prevent the blockage of delay-sensitive flows, their priority can be raised with the use of a priority queue scheduler.

A well-designed congestion control algorithm can also help prevent queues from accumulating, thus directly reducing queuing delay. A new delay-based congestion control algorithm, Copa, has been proposed to achieve low queuing delay and high throughput.[22] It sets an objective function with packet delay as a penalty, and the target sending rate is proportional to the reciprocal of the

queue delay. If the current rate exceeds the target rate, the sender reduces the congestion window, which prevents the accumulation of queues.

Google Congestion Control,[23] an architecture proposed for Web Real-Time Communication (WebRTC), has a delay-based controller to directly manage the rate of the sender. The (one way) queuing delay gradient (the derivative of the queuing delay) serves as the signal to minimize the queuing delay along the end-to-end path. The derivative can reflect the change of the buffer, which provides prescience of buffer size and can be leveraged to reduce latency. Kalman filtering is designed to estimate the queuing delay gradient, and an adaptive threshold of this gradient is set to control the rate of increase or decrease, which helps to minimize the buffer size and queuing delay.

OPPORTUNITIES

Novel applications and scenarios have sprung up that place high requirements regarding latency. In addition, emerging techniques, such as edge computing and data-driven methods, are being applied to reduce latency in many scenarios.

New Applications and Scenarios

An ocean of emerging applications that require ultralow latency have come into existence. One example is VR, which needs low motion-to-photon latency to create the sense of reality. For VR, latency consists of computation time and network transfer time, and the tradeoff between them is under research.

For example, on the basis of the knowledge that VR content rendering can be divided into foreground interactions and predictable background virtual environments, phone–server cooperative rendering has been proposed and proves promising.[24] Foreground rendering is completed at the mobile local GPU. Background prerendering and prefetching are carried out on the server. Rendering the interaction locally can provide a better interaction experience and avoid a long transmission time in the network.

Datacenters are a representative network scenario of low latency. Unlike traditional networks, inside datacenters, architecture and protocols can be designed and modified flexibly for high performance. For instance, existing wireless datacenters face some challenges—e.g. dense interfaces and limited wireless links. A redesign of wireless datacenters using a multireflection ring topology has been proposed.[25] Leveraging a flat reflector, the wireless signal can be reflected several times and transmitted to the target server without traversing multiple hops. This circumvents the queuing and processing delay in intermediate devices.

Furthermore, remote direct memory access (RDMA) supports zero-copy technology and can complete the data transmission without occupying the CPU, permitting low-latency networking. This eliminates the bottleneck of datacenter networks.

Novel Methods to Reduce Latency

There are many novel methods for reducing latency. Fog computing and edge computing put content or services near the user to help reduce physical distance and provide "local computation" capabilities. Delay-sensitive management systems of applications in response to changes of network latency can facilitate the intelligent distribution of content to reduce latency. What's more, using Internet of Things gateways or local processors as the main computing devices for applications avoids the time to pass all information back and forth from the central remote datacenter.

With the development of machine-learning techniques, data-driven methods can facilitate networking optimization and control, including bitrate adaptation, congestion control, and traffic engineering.[26] They can also help achieve low latency. For example, deep learning can be leveraged to solve the topology adaptation problem in datacenter networks, with traffic demand as the input and a near-optimal topology as the output.[27] With an expressive learning framework, this method

can flexibly support optimization over both flow-level or application-level objective functions, including the demand completion time or Hadoop job completion time.

CONCLUSION

Low-latency networking is worth studying and is of critical importance for emerging applications such as VR. This article has presented a short survey on the causes of latency at each layer of the TCP/IP network architecture and different techniques to achieve low latency. We hope that the analysis in this article helps drive the effort forward.

In addition, low-latency networking can enable the development of new infrastructure and new methods. However, challenges and opportunities coexist. We thus encourage the continuous and in-depth study of this problem, which requires combining the competencies of academia and industry.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China project 61872211 and the National Key R&D Program of China (2017YFB1010002). Xin Wang's research is supported by the US National Science Foundation Computer and Network Systems award 1526843.

REFERENCES

1. *Industrial Routing Requirements in Low-Power and Lossy Networks*, K. Pister et al., IETF RFC RFC 5673, IETF, 2009; https://tools.ietf.org/html/rfc5673.
2. *Low Latency Applications and the Internet Architecture*, J. Arkko et al., IETF draft draft-arkko-arch-low-latency-02, IETF, 2017; https://tools.ietf.org/html/draft-arkko-arch-low-latency-02.
3. B. Briscoe et al., "Reducing Internet Latency: A Survey of Techniques and their Merits," *IEEE Comms. Surveys & Tutorials*, vol. 18, no. 3, 2016, pp. 2149–2196.
4. B. Wang et al., "Anatomy of a Personalized Live Streaming System," *Proceedings of the 2016 Internet Measurement Conference* (IMC 16), 2016, pp. 485–498.
5. M Belshe, "More Bandwidth Doesn't Matter (Much)," Google, 8 April 2010; https://goo.gl/PFDGMi.
6. R. Netravali et al., "Polaris: Faster Page Loads Using Fine-grained Dependency Tracking," *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation* (NSDI 16), 2016, pp. 123–136.
7. R. Fielding et al., *Hypertext Transfer Protocol—HTTP/1.1*, R. Fielding et al., IETF RFC RFC 2616, IETF, 1999; https://tools.ietf.org/html/rfc2616.
8. *Hypertext Transfer Protocol Version 2 (HTTP/2)*, M. Belshe et al., IETF RFC RFC 7540, IETF, 2015; https://tools.ietf.org/html/rfc7540.
9. J. Wang et al., "Adaptive media playout buffer management for latency optimization of mobile live streaming," *2017 IEEE International Conference on Multimedia & Expo Workshops* (ICMEW 17), 2017, pp. 369–374.
10. *TCP Fast Open*, Y. Cheng et al., IETF RFC RFC 7413, IETF, 2014; https://tools.ietf.org/html/rfc7413.
11. Y. Cui, "Innovating transport with QUIC: Design approaches and research challenges," *IEEE Internet Computing*, vol. 21, no. 2, 2017, pp. 72–76.
12. "Usage of Site Elements for Websites," Web Technology Surveys; https://w3techs.com/technologies/overview/site_element/all.
13. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, W. Stevens et al., IETF RFC RFC 2001, IETF, 1997; https://tools.ietf.org/html/rfc2001.
14. *Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)*, M. Allman et al., IETF RFC RFC 5827, IETF, 2010; https://tools.ietf.org/html/rfc5827.

15. P. Cheng et al., "Catch the whole lot in an action: Rapid precise packet loss notification in data centers," *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation* (NSDI 14), 2014, pp. 17–28.
16. C. Paasch et al., "Experimental evaluation of multipath TCP schedulers," *Proc. 2014 ACM SIGCOMM Workshop on Capacity Sharing* (CSWS 14), 2014, pp. 27–32.
17. H. Shi et al., "STMS: Improving MPTCP Throughput Under Heterogeneous Networks," *USENIX Annual Technical Conference* (ATC 18), 2018, pp. 719–730.
18. Z. Xu et al., "Experience-driven networking: A deep reinforcement learning based approach," *IEEE Infocom*, 2018.
19. ViewQwest Content Hub, blog; http://blog.viewqwest.com/what-is-latency-based-routing.
20. D. Skordoulis et al., "IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *IEEE Wireless Communications*, vol. 15, no. 1, 2008, pp. 40–47.
21. M. Alizadeh et al., "pFabric: Minimal Near-Optimal Datacenter Transport," *Proceedings of the ACM SIGCOMM 2013 conference* (SIGCOMM 13), 2013; doi.org/10.1145/2486001.2486031.
22. V. Arun et al., "Copa: Practical Delay-Based Congestion Control for the Internet," *15th Usenix Symp. on Networked Systems Design and Implementation*, 2018.
23. G. Carlucci et al., "Analysis and design of the google congestion control for web real-time communication (WebRTC)," *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, p. 13.
24. Z. Lai et al., "Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices," *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017.
25. Y. Cui et al., "Diamond: Nesting the data center network with wireless rings in 3-d space," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, 2018, pp. 145–160.
26. M. Wang et al., "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, 2018, pp. 92–99.
27. M. Wang et al., "Neural Network Meets DCN: Traffic-driven Topology Adaptation with Deep Learning," *Proceedings of the ACM on Measurement and Analysis of Computing Systems 2.2*, 2018, p. 26.

ABOUT THE AUTHORS

Xutong Zuo is a PhD student in Tsinghua University's Department of Computer Science and Technology. Her research interests include low-latency networking and live streaming. Zuo received a BE in computer science and technology from the Beijing University of Posts and Telecommunications. Contact her at zuoxt18@mails.tsinghua.edu.cn.

Yong Cui is a full professor in Tsinghua University's Department of Computer Science and Technology. His research interests include computer network architecture and mobile computing. Cui received a PhD in computer science from Tsinghua University. He's the corresponding author. Contact him at cuiyong@tsinghua.edu.cn.

Mowei Wang is working toward his PhD in Tsinghua University's Department of Computer Science and Technology. His research interests are in datacenter networks and machine learning. Wang received a BE in communication engineering from the Beijing University of Posts and Telecommunications. Contact him at wang.mowei@outlook.com.

Tianxiao Xiao is an undergraduate student in Syracuse University's Department of Engineering and Computer Science. His research interests are in machine learning and networking. Contact him at williamshawxtx@gmail.com.

Xin Wang is an associate professor in Stony Brook University's Department of Electrical and Computer Engineering. Her research interests include mobile computing and wireless networking. Wang received a PhD in electrical and computer engineering from Columbia University. Contact her at x.wang@stonybrook.edu.

Contact department editor Yong Cui at cuiyong@tsinghua.edu.cn.

On Routing and Forwarding

Vinton G. Cerf
Google

Everything on the Internet need not be connected.

In the earliest days of Internet design, I was largely fixated on the idea that everything should be able to talk to everything else on the network. A recipient of an Internet packet could reject or ignore it but senders were free to send. As the Internet penetrated into the commercial private sector, enterprises looked for ways to isolate their computing equipment from the global network through the use of firewalls that were not part of the original design.

In subsequent years, routers have become extremely elaborate systems and routing methods have become more complex. The Border Gateway Protocols grew more complex as routing choices collided with economic considerations (e.g. near-end vs. far-end hop off). Multiprotocol Label Switching (MPLS) was used to groom traffic onto alternative optical streams to manage channel occupancy. Virtual LANs were developed to group devices together in a common communication channel even when they were on distinct physical LANs. OpenFlow expanded the basis for routing and forwarding decisions from simple destination address lookup to use of any bits in a packet on which to base forwarding choices. It also demonstrated the feasibility of centralizing forwarding table production for systems of suitable scale. The core, inter-data-center network of Google adopted this practice to very good effect.

It has finally dawned on me that not everything has to be connected to everything and that routing and forwarding can be deliberately constructed to confine connectivity to a desired cohort of devices. I have been so consumed with "everything has to be connected" that this recognition has been slow in coming. Duh. The forwarding tables in routers can be constructed in many ways and more than one forwarding table can readily be imagined. The implication is that one can use conventional routers or more recent Software Defined Networks (SDNs) to isolate groups of devices from the rest of the Internet. Such an implementation strikes me as a different way to realize the concept of Virtual Private Network without the encapsulation and potential hazards of conventional VPN implementation. The forwarding table defines the constituents of the virtual network. A flat table of specific 32 bit or 128 bit IP addresses could create an isolated group of devices able to communicate only with each other, for example.

Interestingly, the redefinition of forwarding tables also leads to concepts such as Information Centric Networks or Content Centric Networks that route on content indicators rather than addresses. Moreover, one can extend this line of thinking in other ways. The existing Domain Name System (DNS), for example, maps domain names into IP addresses which are then routed by the common routing system(s) of the Internet. It is quite possible to imagine a different set of identifiers, other than domain names, that could be mapped into IP addresses. One example is the Digital Object Architecture developed by Robert Kahn at the Corporation for National Research

Published by the IEEE Computer Society
1089-7801/18/$33.00 ©2018 IEEE

Initiatives (CNRI). Every digital object gets a unique digital identifier (handle) that can be looked up to find the location or locations at which the object may be found. David D. Clark's forthcoming book, Designing an Internet (Information Policy), explores a variety of potential new designs for Internet-like functionality.

As memory becomes less a barrier and backbone link capacities increase, more elaborate routing mechanisms may prove feasible. New SDN designs, with switches that are programmable, may provide a basis for more refined and sophisticated routing and forwarding mechanisms. In the coming era of the Internet of Things, the ability to isolate groups of devices for protective reasons may prove to be an essential step towards improving the security of the Internet. Such ideas will place significant demand on configuration tools to cope with scale and the use of cryptography to protect against the risks of misrouting.

I am sure that there are among the readers of this column, many who are much more cognizant of advanced thinking about routing and forwarding, so I hope they will take time to draw attention to their ideas for future evolution of this all-important function of the Internet.

BIO

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 Internet Computing

EDITORIAL OFFICE
Publications Coordinator: internet@computer.org
Authors: www.computer.org/web/peer-review/magazines
Letters to the Editors: bbrannon@computer.org
Subscribe: www.computer.org/subscribe
Subscription change of address:
address.change@ieee.org
Missing or damaged copies: help@computer.org
Reprints of articles: internet@computer.org
IEEE Internet Computing
c/o IEEE Computer Society
10662 Los Vaqueros Circle, Los Alamitos, CA 90720 USA
Phone +1 714 821 8380; Fax +1 714 821 4010
www.computer.org/internet-computing

Circulation: *IEEE Internet Computing* (ISSN 0272-1716) is published bimonthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; voice +714 821 8380; fax +714 821 4010; IEEE Computer Society Headquarters, 1828 L St. NW, Suite 1202, Washington, DC 20036. Visit www.computer.org/subscribe for subscription information.
Postmaster: Send undelivered copies and address changes to *IEEE Internet Computing*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the 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 the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space. IEEE prohibits discrimination, harassment, and bullying: For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.