Edge Computing

**EXPERT-CURATED
GUIDES TO
THE BEST OF
CS RESEARCH**

NITESH MOR

Research for Practice combines the resources of the ACM Digital Library, the largest collection of computer science research in the world, with the expertise of the ACM membership. In every RfP column experts share a short curated selection of papers on a concentrated, practically oriented topic.

Cloud computing, a term that elicited significant hesitation and criticism at one time, is now the de facto standard for running always-on services and batch-computation jobs alike. In more recent years, the cloud has become a significant enabler for the IoT (Internet of things). Network-connected IoT devices—in homes, offices, factories, public infrastructure, and just about everywhere else—are significant sources of data that need to be handled and acted upon. The cloud has emerged as an obvious support platform because of its cheap data storage and processing capabilities, but can this trend of relying exclusively on the cloud infrastructure continue indefinitely?

For the applications of tomorrow, computing is moving out of the silos of far-away datacenters and into everyday lives. This trend has been called *edge computing*, *fog computing*, *cloudlets*, as well as other designations. In this article, edge computing serves as an umbrella term for this trend. While cloud computing infrastructures proliferated because of flexible pay-as-you-go economics and the ability to outsource resource management, edge computing is a growing trend to satisfy the needs for lower latency, higher bandwidth, and improved reliability to support richer applications. Plus, both privacy concerns

and legislation that require data to be confined to a specific physical infrastructure are also driving factors for edge computing.

It is important to note that edge computing is not merely caching, filtering, and preprocessing of information using on-board resources at the source/sink of data—the scope of edge computing is much broader. Edge computing also includes the use of networked resources closer to the sources/sinks of data. In an ideal world, resources at the edge, in the cloud, and everywhere in between form a continuum. Thus, for power users such as factory floors, city infrastructures, corporations, small businesses, and even some individuals, edge computing means making appropriate use of on-premises resources together with their current reliance on the cloud. In addition to existing cloud providers, a large number of smaller but more optimally located service providers are likely to flourish and handle the overflow demand from power users as well as support novice users.

Creating edge computing infrastructures and applications encompasses quite a breadth of systems research. Let's take a look at the academic view of edge computing and a sample of existing research that will be relevant in the coming years.

A VISION FOR EDGE COMPUTING: OPPORTUNITIES AND CHALLENGES

Let's start with an excellent paper that introduced the term *fog computing* and highlights why practitioners should care about it:

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. 2012. Fog computing and its role in the Internet of things. In Proceedings of the First Edition of the Workshop on Mobile Cloud Computing, 13-16; ACM. <https://dl.acm.org/citation.cfm?id=2342513>

Although short, this paper provides a clear characterization of fog computing and a concise list of the opportunities it provides. It then goes deeper into the discussion of richer applications and services enabled by fog computing, such as connected vehicles, smart grid, and wireless sensor/actuator networks. The key takeaway is that the stricter performance/QoS requirements of these rich applications and services need: (a) better architectures for compute, storage, and networking; and (b) appropriate orchestration and management of resources.

While this paper is specifically about the Internet of things and fog computing, the same ideas apply to edge computing in the broader sense. Unsurprisingly, the opportunities of edge computing also come with a number of challenges. An in-depth case study of some of these challenges and possible workarounds is illustrated in the [FarmBeats](#) project that was featured in the November/December 2017 installment of RfP, “Toward a Network of Connected Things.”

A WORLD FULL OF INFORMATION: WHY NAMING MATTERS

One of the hurdles in using resources at the edge is the complexity that they bring with them. What can be done to ease the management complexity? Are existing

architectures an attempt to find workarounds for some more fundamental problems?

ICNs (information-centric networks) postulate that most applications care only about information, but the current Internet architecture involves shoehorning these applications into a message-oriented, host-to-host network. While a number of ICNs have been proposed in the past, a recent notable paper addresses NDN (named data networking).

Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K.C., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. 2014. [Named data networking](https://dl.acm.org/citation.cfm?id=2656887). ACM SIGCOMM Computer Communication Review 44(3), 66-73; <https://dl.acm.org/citation.cfm?id=2656887>

NDN, like many other ICNs, considers named information as a first-class citizen. Information is named with human-readable identifiers in a hierarchical manner, and the information can be directly accessed by its name instead of through a host-based URL scheme.

As for the architecture of the routing network itself, NDN has two types of packets: interest and data. Both types are marked with the name of the content. A user interested in specific content creates an interest packet and sends it into the network. The NDN routing protocol is based on a name-prefix strategy, which, in some ways, is similar to prefix aggregation in IP routing. An NDN router, however, differs from IP routers in two important ways: (a) it maintains a temporary cache of data that it has seen so far, so that any new interests from downstream nodes

can be responded to directly without going to an upstream router; and (b) only one request is sent to the upstream router for multiple interests to the same name by a number of downstream nodes. Multiple paths for the same content are also supported.

The security in NDN is also data-centric. Each data packet is cryptographically signed by the producer of the data, and a consumer can reason about data integrity and provenance from such signatures. In addition, encryption of data packets can be used to control access to information.

Using human-readable names allows for the creation of predictable names for content, which is useful for a certain class of applications. The paper also describes how applications would look with NDN by using a number of examples such as video streaming, realtime conferencing, building automation systems, and vehicular networking.

NDN is not the first ICN, and it isn't the last. Earlier ICNs were based on flat cryptographic identifiers for addresses, compared to NDN's hierarchical human-readable names. A more detailed overview of ICNs, their challenges, commonalities, and differences can be found in a 2011 survey paper by Ghodsi et al. ([Information-centric networking: Seeing the forest for the trees](#)).

To provide a little historical context, NDN was one of several FIAs (future Internet architectures) funded by the National Science Foundation. It is instructive to look at a few other projects, such as [XIA](#) and [MobilityFirst](#), that share the goals of cleaner architectures for the future Internet.

The key lesson for practitioners is that choosing the right level of abstractions is important for ensuring

appropriate separation of concerns between applications and infrastructure.

SECURING EXECUTION

While information management is important, let's not forget about computation. Whereas cryptographic tools can help secure data, it is equally important to secure the computation itself. As containers have risen in popularity as a software distribution and lightweight execution environment, it is important to understand the security implications—not only for isolation among users, but also for protection from platform and system administrators.

*Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'Keeffe, D., Stillwell, M. L., Goltzsche, D., Eysers, D., Kapitza, R., Pietzuch, P., Fetzer, C. 2016 (November). [SCONE: Secure Linux Containers with Intel SGX](https://www.usenix.org/system/files/conference/osdi16/osdi16-arnautov.pdf). In *Operating Systems Design and Implementation 16*, 689-703; <https://www.usenix.org/system/files/conference/osdi16/osdi16-arnautov.pdf>*

SCONE implements secure application execution inside *Docker* using *Intel SGX*, assuming one trusts Intel SGX and a relatively small TCB (trusted computing base) of SCONE. Note that system calls cannot be executed inside an SGX enclave itself and require expensive enclave exits. The ingenuity of SCONE is in making existing applications work with acceptable performance without source code modification, which is important for real-world adoption.

While this paper is quite detailed and instructive, here is a very brief summary of how SCONE works. An application

is compiled against the SCONE library, which provides a C standard library interface. The SCONE library provides “shielding” of system calls by transparently encrypting/decrypting application data. To reduce the performance degradation, SCONE also provides a user-level threading implementation to maximize the time threads spend inside the enclave. Further, a kernel module makes it possible to use asynchronous system calls and achieve better performance; two lock-free queues handle system call requests and responses, which minimizes enclave exits.

Integration with Docker allows for easy distribution of packaged software. The target software is included in a Docker image, which may also contain secret information for encryption/decryption. Thus, Docker integration requires protecting the integrity, authenticity, and confidentiality of the Docker image itself, which is achieved with a small client that is capable of verifying the security of the image based on a startup configuration file. Finally, the authors show that SCONE can achieve at least 60 percent of the native throughput for popular existing software such as Apache, Redis, and memcached.

While technologies such as Intel SGX do not magically make applications immune to software flaws (as has been demonstrated by *Spectre* and *Foreshadow*), hardware-based security is a step in the right direction. Computing resources on the edge may not have physical protections as effective as those in cloud data centers, and consequently, an adversary with physical possession of the device is a more significant threat in edge computing.

For practitioners, SCONE demonstrates how to build a practical secure computation platform. More importantly,

SCONE is not limited to edge computing; it can also be deployed in existing cloud infrastructures and elsewhere.

A UTILITY PROVIDER MODEL OF COMPUTING

Commercial offerings from existing service providers, such as Amazon's AWS IoT GreenGrass and AWS Snowball Edge, etc., enable edge computing with on-premises devices and interfaces that are similar to existing cloud offerings. While using familiar interfaces has some benefits, it is time to move away from a "trust based on reputation" model.

Is there a utility-provider model that provides verifiable security without necessarily trusting the underlying infrastructure or the provider itself? Verifiable security not only makes the world a more secure place, it also lowers the barrier to entry for new service providers that can compete on the merits of their service quality alone.

The following paper envisions a cooperative data utility model where users pay a fee in exchange for access to persistent storage. Note that while many aspects of the vision may seem like a trivial task with the cloud computing resources of today, this paper predates the cloud by almost a decade.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. 2000. [OceanStore: an architecture for global-scale persistent storage](https://dl.acm.org/citation.cfm?id=357007). ACM SIGOPS Operating Systems Review 34(5), 190-201; <https://dl.acm.org/citation.cfm?id=357007>

OceanStore assumes a fundamentally untrusted infrastructure composed of geographically distributed servers and provides storage as a service. Data is named by GUIDs (globally unique identifiers) and is portrayed as nomadic (i.e., it can flow freely and can be cached anywhere, anytime). The underlying network is essentially a structured P2P (peer-to-peer) network that routes data based on the GUIDs. The routing is performed using a locality-aware, distributed routing algorithm.

Updates to the objects are cryptographically signed and are associated with predicates that are evaluated by a replica. Based on such evaluation, an update can be either committed or aborted. Further, these updates are serialized by the infrastructure using a primary tier of replicas running a Byzantine agreement protocol, thus removing the trust in any single physical server or provider. A larger number of secondary replicas are used to enhance durability. In addition, data is replicated widely for archival storage by using erasure codes.

While OceanStore has a custom API to the system, it provides “facades” that could offer familiar interfaces—such as a filesystem—to legacy applications. This is a vision paper with enough details to convince readers that such a system can actually be built.

In fact, OceanStore had a follow-up prototype implementation named [Pond](#). In a way, OceanStore can be considered a two-part system: an information-centric network underneath and a storage layer on top that provides update semantics on objects. Combined with the secure execution of Intel SGX-like solutions, it should be possible, in theory, to run end-to-end, secure applications.

Although OceanStore appeared before the cloud was a thing, the idea of a utility model of computing is more important than ever. For practitioners of today, OceanStore demonstrates that it is possible to create a utility-provider model of computing even with a widely distributed infrastructure controlled by a number of administrative entities.

FINAL THOUGHTS

Because edge computing is a rapidly evolving field with a large number of potential applications, it should be on every practitioner's radar. While a number of existing applications can benefit immediately from edge computing resources, a whole new set of applications will emerge as a result of having access to such infrastructures. The emergence of edge computing does not mean that cloud computing will vanish or become obsolete, as there will always be applications that are better suited to being run in the cloud.

This Research for Practice piece merely scratches the surface of a vast collection of knowledge. A key lesson, however, is that creating familiar gateways and providing API uniformity are merely facades; infrastructures and services are needed to address the core challenges of edge computing in a more fundamental way.

Tackling management complexity and heterogeneity will probably be the biggest hurdle in the future of edge computing. The other big challenge for edge computing will be data management. As data becomes more valuable than ever, security and privacy concerns will play an important role in how edge computing architectures and

applications evolve. In theory, edge computing makes it possible to restrict data to specific domains of trust for better information control. What happens in practice remains to be seen.

Cloud computing taught practitioners how to scale resources within a single administrative domain. Edge computing requires learning how to scale in the many administrative domains.

Nitesh Mor is a Ph.D. candidate in computer science at the University of California, Berkeley, where he is advised by John Kubiatowicz. He is currently part of the Global Data Plane project at the Ubiquitous Swarm Lab at UC Berkeley, where he focuses on secure Internet-wide infrastructures for data storage and communication. More broadly, his research focuses on data-oriented middleware, which intersects with security, privacy, systems, and networks in general. Previously, he has worked on privacy-preserving web-search; secure, fair-exchange protocols; and micropayment architectures.

Copyright © 2018 held by owner/author. Publication rights licensed to ACM.