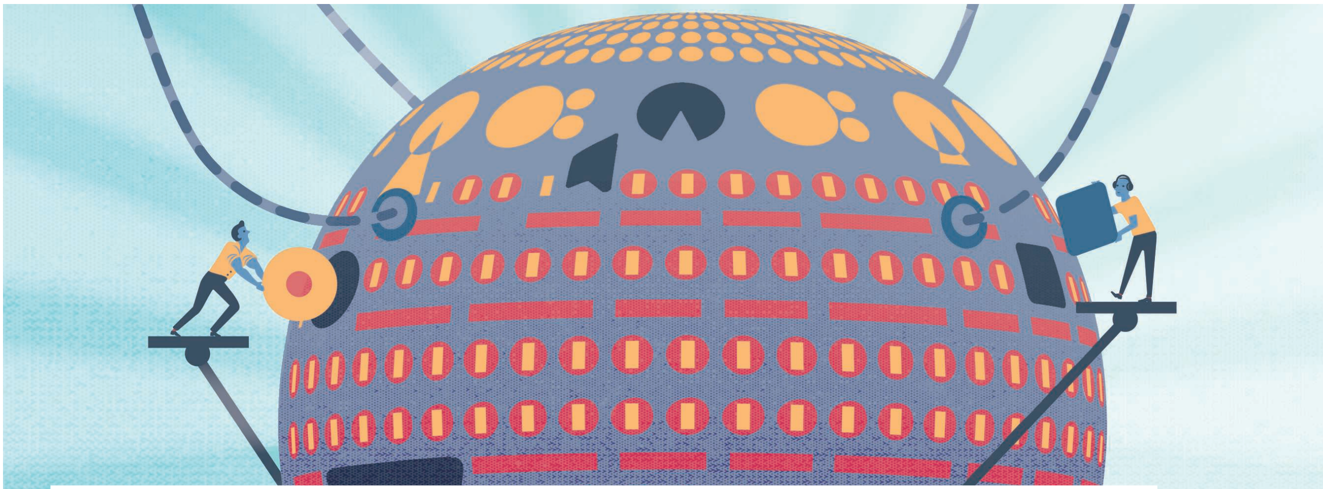


Creating a Resilient IoT With Edge Computing

Hokeun Kim and Edward A. Lee, University of California, Berkeley
Schahram Dustdar, Technische Universität Wien

Denial-of-service (DoS) attacks on the safety-critical Internet of Things (IoT) can lead to life-threatening consequences, and the risk of these attacks is increasing. We propose levels of context awareness to address availability threats and illustrate how context-aware edge computing enhances the IoT's resilience to DoS attacks through our edge-computing-based security solution.

Availability can be safety critical for many Internet of Things (IoT) systems. In October and November of 2016, a distributed denial-of-service (DDoS) attack on building control systems shut down the heating systems of buildings in Finland for more than a week.[1] Winter is harsh in Finland; therefore, failures in heating systems raise safety concerns, not just inconvenience. Meanwhile, the risk of DDoS attacks almost doubled in 2017 compared with 2016 due to the increasing number of IoT devices, according to a report by Corero Network Security.[2] An illustrative example of DDoS launched by the IoT was the Mirai botnet,[3] which compromised hundreds of thousands of IoT devices and attacked dynamic domain name system (DNS) servers, disrupting Internet connections to major websites.

Compromising the availability of computer systems is the main goal of DoS attacks. DDoS attacks are one category of DoS attacks that send excessive network traffic to the target servers to exhaust their computational and communicational resources. For the IoT, including cyberphysical

Digital Object Identifier 10.1109/MC.2018.2888768
Date of publication: 30 July 2019

systems (CPSs) interacting with humans and the physical world, availability attacks, such as DDoS attacks, can pose safety threats. Therefore, it is crucial to build IoT systems that are resilient to availability threats.

However, many current IoT solutions are built around cloud computing, which makes the system's availability dependent on remote cloud servers. Google's OnHub incident[4] demonstrated the risk of depending on remote servers for the IoT. On 23 February 2017, Google's smart routers, called *OnHub*, suddenly stopped working, and the IoT devices connected to the routers also became unavailable because Google's remote authentication servers failed.

Devices at the network edge, called *edge computers*, are becoming smarter and more capable. Edge computing,[5] which uses edge computers to offer computational and communicational resources for IoT devices (referred to as *things* in this article), brings opportunities to build resilient IoT systems. In this article, we propose an authentication and authorization infrastructure for the IoT based on context-aware edge computing, along with levels of context awareness for edge computers for building resilient IoT systems.

PERSPECTIVES OF THE IOT

Different IoT perspectives can lead to different system designs. This section compares two ways of viewing the IoT to highlight the importance of considering an underlying network architecture when designing resilient IoT systems.

Cloud-centric perspective

A *cloud-centric perspective* of the IoT is a conceptual view that considers the cloud as a central platform for the IoT and edge computers as the edge of the cloud, as shown in Figure 1(a). This perspective is widely adopted, for example, in fog computing,[6] where the cloud comprises core services and the edge is local proxies for the cloud, mainly for offloading part of the cloud's workload. From this perspective, edge computers play supportive roles for IoT services and applications. Cloud computing-based IoT solutions[7] use cloud servers for various purposes, such as massive computation, data storage, communication between IoT systems, and security. However, the cloud-centric perspective misses important facts in the real network architecture of the IoT.

- In the network architecture, the cloud is also located at the network edge, not surrounded by the edge.
- Computers at the edge do not always have to depend on the cloud; they can operate autonomously and collaborate with one another directly.

Internet-centric perspective

To better discuss how to protect IoT systems from DoS threats, we propose a new perspective called the *Internet-centric perspective*, shown in Figure 1(b), which views the Internet as a center of the IoT architecture and considers the edge components as gateways to the Internet, not to the cloud. Each local network can be organized around the edge computers autonomously. Thus, the local systems are not always dependent on the cloud or the Internet. The Internet-centric perspective captures essential aspects of the IoT.

- Things in the IoT belong to partitioned subsystems or local networks rather than to a big centralized system.
- The cloud is also connected to the Internet via the edge of the network.
- Remote IoT systems can be connected directly via the Internet, and their communication does not have to go through the cloud.
- The edge can connect things to the Internet and also disconnect the traffic from outside to protect things. For this, the local IoT system must be able to operate autonomously, although the system's performance might be affected once it loses the cloud's support.

TO BETTER DISCUSS HOW TO PROTECT IoT SYSTEMS FROM DoS THREATS, WE PROPOSE A NEW PERSPECTIVE CALLED THE *INTERNET-CENTRIC PERSPECTIVE*.

TOWARD RESILIENT IOT SYSTEMS

With this Internet-centric perspective, how can we build resilient IoT systems? We can start with the tactics that attackers would likely take to cause DoS.

Of course, the attackers can target the cloud to exhaust the resources of

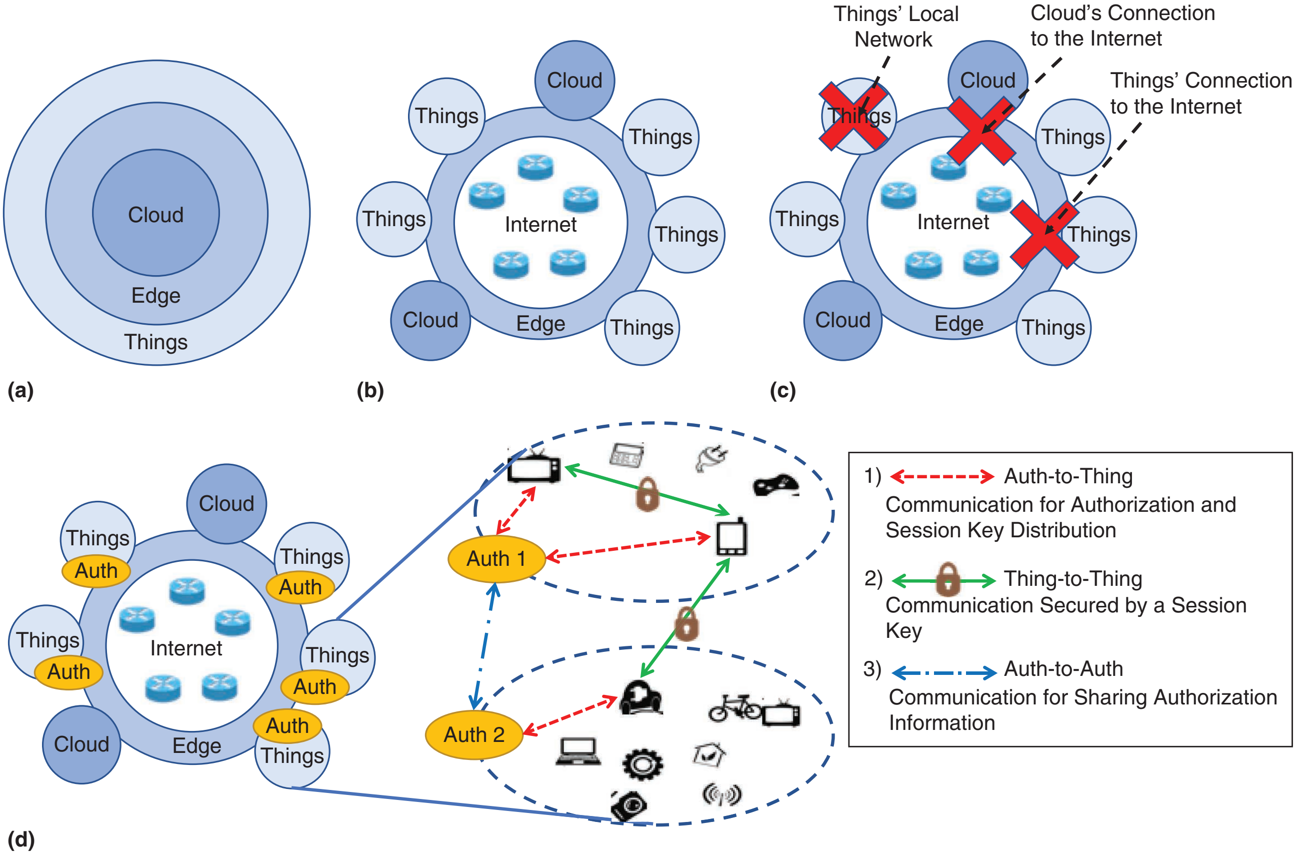

Cloud
Edge
Things
(a)
Cloud
Things
Things
Things
Internet
Things
Cloud
Edge
Things
(b)
Things' Local Network
Cloud's Connection to the Internet
Things' Connection to the Internet
Cloud
Things
Things
Things
Internet
Things
Cloud
Edge
Things
(c)
Cloud
Things
Auth
Things
Auth
Things
Auth
Internet
Things
Auth
Auth
Things
Cloud
Edge
Auth 1
Auth 2
1) Auth-to-Thing Communication for Authorization and Session Key Distribution
2) Thing-to-Thing Communication Secured by a Session Key
3) Auth-to-Auth Communication for Sharing Authorization Information
(d)

FIGURE 1. The different perspectives for IoT and SST architectures. (a) A cloud-centric perspective: the edge is depicted as the edge of the cloud. (b) An Internet-centric perspective: the edge is depicted as the edge of the Internet. (c) Vulnerable spots can be exploited to cause DoS. (d) SST architecture features locally centralized and globally distributed Auths for authentication and authorization. Auth: authentication and authorization entity.

cloud servers. However, cloud servers are inside data centers and well protected against availability attacks from the outside by many layers of defense, including firewalls. It would be very challenging for the attackers to take down a single data center, even with a lot of resources and effort. Moreover, many commercial cloud services consist of globally distributed data centers, making it even more difficult to take down all of the data centers.

Alternatively, the attackers could try other approaches. Figure 1(c) shows weak points that attackers could exploit to cause DoS in IoT systems without directly attacking the cloud. As long as the attackers can disrupt the IoT systems, they succeed. The attackers can hamper the connection between the cloud and things, for example, by making DNS services unavailable, as the Mirai botnet did. They can also attack the local network to disrupt the IoT services directly. Therefore, it is not enough to protect the cloud servers; the individual local IoT networks also need to be protected.

There are a couple of fundamental requirements for resilient IoT systems. First, we must be prepared for when the cloud is not available. The IoT systems should be able to provide at least vital services, for example, the heating systems in cold regions, even when the cloud is not available. In this sense, we can use edge computers as local controllers for things as a backup for the cloud. In general, edge computers have more resources than things and can be local central points.

Second, a local IoT system should be equipped with defense mechanisms against availability threats, including detecting and mitigating the impact of attacks and reacting to failures to recover the system's availability. Edge computers can play key roles in implementing such defense mechanisms. For example, since an edge computer sits between things and the Internet, it can detect an incoming DDoS attack and protect the local network by blocking the external traffic. If edge computers are aware of the local system's characteristics and how the system should behave, for example, the expected volume of data traffic or desirable temperature ranges, they can detect the anomaly and recover the normal state. Edge computers can also use local resources and security measures to recover availability.

A RESILIENT SECURITY SOLUTION FOR THE IOT

We present our open source edge-computing-based IoT security solution, which is resilient to availability threats. Called the *Secure Swarm Toolkit* (*SST*),[8] it is freely available at https://github.com/iotauth. The SST provides authentication and authorization services for the IoT. *Authentication* is a process of identifying devices or users, and *authorization* is a process of controlling access to important resources, such as the control of CPSs. These two processes are critical for security, safety, and availability, as shown in Google's OnHub incident, where authentication problems of Google servers led to the entire system's failure.

Resilient edge-computing-based architecture

Figure 1(d) illustrates the SST's architecture, which is locally centralized and globally distributed,[9] and has many potential advantages in building resilient IoT systems. In the SST, things are authenticated and authorized by an edge-hosted locally centralized entity called *Auth*.[10] Auths authenticate and authorize local things [number 1 in Figure 1(d)] and provide them with session keys for securing thing-to-thing communication [number 2 in Figure 1(d)]. By running security functions on the edge, the IoT systems can continue authentication and authorization processes even when cloud servers are unavailable. Moreover, Auths monitor the entire access activity among things, allowing them to detect an anomaly in the system. Auths can also protect the local IoT networks from external attackers, using defense mechanisms, such as firewalls, and physically disconnecting the DDoS traffic toward local systems. The locally centralized architecture enables Auths to react to compromised things in a timely way.

The globally distributed architecture of the SST makes the IoT systems scalable[8] and also enhances the resiliency of the IoT.[11] Auths share authorization information and session keys to authorize things in different networks [number 3 in Figure 1(d)]. Auths hosted on edge computers will be more geographically distributed than cloud servers, making it even more challenging for attackers to take down the IoT system by attacking edge computers. The cloud servers in data centers may become unreachable by disrupting DNS services or the Internet connection to data centers. However, this type of attack will be less effective for the SST because many Auths will be reachable through local networks even when the Internet connection is unstable.

CONTEXT AWARENESS OF THE CLOUD AND EDGE

In computing, *context* has various meanings. Here, we consider *context* as information about the environments in which the IoT systems operate, including underlying platforms, available devices, network topology, location, and time. *Context awareness* refers to the capability of computers in the IoT to sense and react to what is happening in their operating environments. This is crucial for the resilience of an IoT system because it enables the computers in the system to mitigate the impact of an attack and recover from a failure. Context awareness has been related to security of the IoT, and examples include using it for trust initialization[12] and trust management.[13] In the IoT, the cloud and the edge will have different types of context awareness.

Global context in the cloud

The cloud will have a better holistic view of global context than the edge. In the smart city example in Figure 2, we assume that the subsystems are connected to the cloud, which receives real-time data from them. Cloud servers will have a better understanding of what is happening in the city and which subsystem has problems, for example, whether the power failure in manufacturing systems is due to a failure in power plants. The cloud will also be able to control the subsystems and react to possible incidents using *global context awareness*. When the air quality measured by an environmental monitoring system is not healthy, the cloud can order smart buildings to close windows and activate air purifiers. For traffic infrastructure, the cloud can monitor the traffic situation and control traffic signals to ease a traffic jam.

Local context in the edge

The edge will have a better awareness of *local context*. Edge computers will

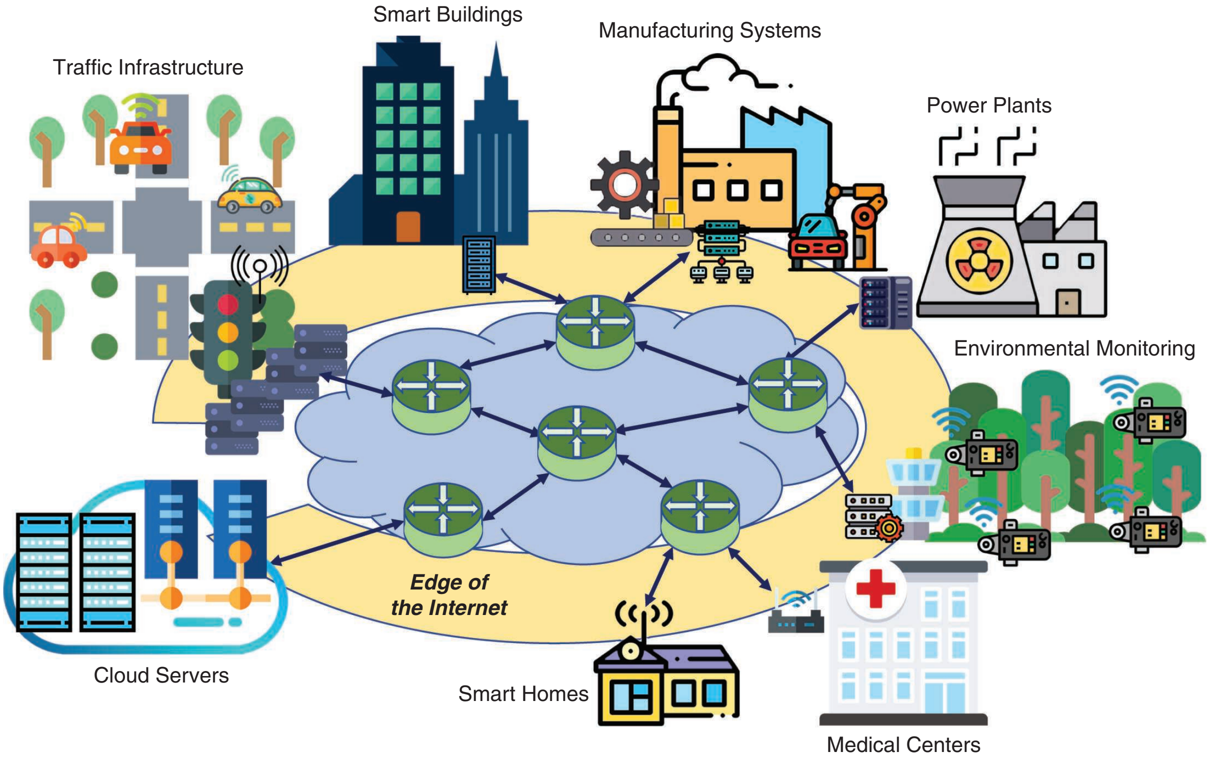


FIGURE 2. A smart city with the edge of the Internet.

have access to the raw communication packets and data within local IoT systems. For example, edge computers for the traffic infrastructure will be aware of the real-time video data at crossroads, and those for environmental monitoring can analyze raw sensor data. Also, edge computers managed by the local administrators can access data not available to external systems for privacy reasons, such as the on-body monitoring data of medical centers or surveillance camera data of smart homes. The edge can view incoming and outgoing data from the local system; thus, it will be able to detect DDoS attacks toward the IoT system, which can be challenging for the cloud. Edge computers will be better aware of the network topology and locally available resources that can be used to mitigate threats and recover availability. Thanks to their proximity to local systems, edge computers will be able to detect availability threats and take better actions in a more timely fashion than remote cloud servers.

The cloud and the edge have different types of context awareness that complement each other. The global context awareness of the cloud fosters collaboration between heterogeneous subsystems, whereas the local context awareness of the edge enables subsystem-specific analysis and close interaction with things. We focus on the local context awareness of the edge with regard to building robust IoT systems.

CONTEXT AWARENESS AND RESILIENCE

Local context awareness is especially important for resilience. We propose five awareness levels for the edge computing-based IoT: event, situation, adaptability, goal, and future awareness (Table 1).

Event awareness is the simplest capability for sensing and monitoring environments. An event-aware system can react to sensed events according to predefined rules. Examples of defense mechanisms and tools used by event-aware systems include firewalls and rule engines. This level provides data filtering and dissemination as infrastructural services to higher awareness levels.

Situation awareness is a more advanced capability for understanding the implication of a series of events and reacting to the situation based on understanding. For example, network intrusion detection systems not only monitor the network traffic but also detect network intrusion by analyzing characteristics or anomalies of the data traffic. Situation-aware systems often use statistical tools to detect anomalies.

A system with *adaptability awareness* can change and modify itself if necessary when the system detects threats or failures. Adaptability awareness includes knowledge and control of the available resources and how the resources can be used to recover and maintain availability even under failures. Many reconfigurable systems will have this level of awareness, including software-defined networking. This level of awareness makes IoT systems more resilient even when some of the important components, for example, the edge computers, become unavailable.

Goal awareness introduces goals expressing the overall objectives and purposes of a self-adaptive system. When there are multiple goals, an IoT system must be able to resolve conflicting goals within resource constraints. Therefore, a goal-aware system can take priorities and tradeoffs into account when adapting to new situations. Such systems include mixed-criticality systems that contain tasks with different criticality levels on a single platform.

Future awareness is an ultimate form of awareness that enables self-sustainable IoT systems. A future-aware system is capable of predicting longer-term effects of short-term adaptation actions and considering future resource

TABLE 1. Awareness levels in IoT systems.

Awareness levels	Characteristics	Capabilities	Examples
Event	A system collects simple events that trigger basic event-condition-action rules. The system has no explicit knowledge of the resources needed nor whether the adaptation has a long-lasting (positive) effect.	To react to events based on predefined rules, regardless of any other factors in the situation	Firewalls, rule engines (e.g., Drools), IFTTT.com (if this then that)
Situation	The ability to perceive the status of a system by aggregating relevant events. The system understands the implication of individual events in a greater context.	To react to events properly in context, with the capability to collect and understand local contextual information	Network intrusion detection systems, anomaly detection systems
Adaptability	The awareness of the possible adaptation capability of a system in its environment. At this level, cooperative adaptation can be conducted spontaneously based on the knowledge of adaptability.	To initiate spontaneous collaboration with other edge computers and controllable environmental conditions	Reconfigurable software-defined networking, reconfigurable CPSs
Goal	The awareness of the goals of a system as a whole. In IoT systems, a goal includes not only the desired functionality of a service but also nonfunctional properties and resource constraints imposed by the environment. In the presence of conflicting goals, systems with this level of awareness also consider the potential tradeoffs and priorities (criticalities) among goals.	To negotiate with other edge computers regarding resource allocation To understand the significance of a potential failure and attempt to avoid it accordingly	Mixed-criticality systems (e.g., avionic systems, autonomous vehicles)
Future	The awareness of a system's lifecycle describing long-term utilization and resource provisioning by the environment. This requires information on the probable future system state based on scheduled or expected future events. Ultimately, this level describes systems that can select appropriate short-lived adaptation actions that respect long-term resource constraints and goals.	To predict resource consumption, user behavior, and future resource requirements To act according to predictions	Self-sustainable smart city

provisions and constraints when utilizing short-term resources. For example, edge computers in a future-aware system should be able to anticipate wear-out and replacement cycles of things, such as battery-powered sensor nodes, and take proper actions to maintain long-term availability.

In summary, the awareness levels help us understand what the IoT system should know to support a certain level of resilience. Event awareness is the most fundamental level of awareness that needs to be part of higher levels. To reach the future-awareness level, an IoT system will require all lower levels of awareness.

CONTEXT-AWARE EDGE COMPUTING FOR A RESILIENT IOT

Even with the SST's architectural advantages, there are still availability threats to edge-computing-based IoT systems. To cause DoS in such systems, attackers will probably target edge computers rather than individual things, to maximize the impact of an attack.

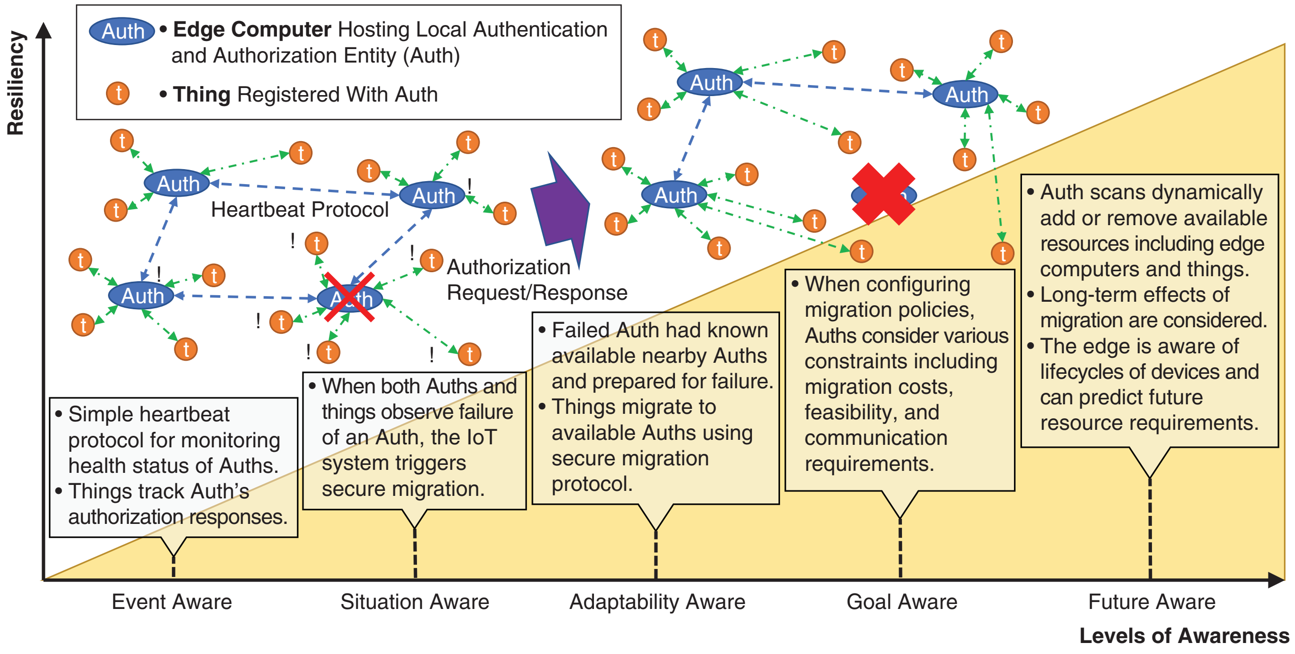
In distributed systems, it is common to replicate resources across distributed computers to increase availability. Such systems include content delivery networks, for example, Akamai and Limelight Networks. However, distributing authentication- and authorization-related information is trickier than sharing content resources, such as web pages, images, or videos, because we need to consider trust among things and edge computers.

Secure migration and awareness levels

Auths in the SST maintain distributed trust relationships with one another. This allows other trusted Auths to take over authentication and authorization services for things in the case of their Auth's failure. We call this technique *secure migration*, in which the things migrate from unavailable Auths to other trusted Auths to continue security services while keeping trust relationships intact.

Figure 3 demonstrates how the SST's secure migration technique implements context-awareness levels to mitigate availability threats. The SST's implementation provides practical insights for considerations in building a resilient IoT using edge computing at all five levels of awareness.

Event awareness. In SST, edge computers (Auths) and things use simple mechanisms to sense network conditions and the health status of the system. Auths check each other periodically using a heartbeat protocol. Things send authentication


Resiliency
Auth • Edge Computer Hosting Local Authentication and Authorization Entity (Auth)
t • Thing Registered With Auth
Auth
t
Heartbeat Protocol
Authorization Request/Response
!
Event Aware
Situation Aware
Adaptability Aware
Goal Aware
Future Aware
Levels of Awareness

- Simple heartbeat protocol for monitoring health status of Auths.
- Things track Auth's authorization responses.

- When both Auths and things observe failure of an Auth, the IoT system triggers secure migration.

- Failed Auth had known available nearby Auths and prepared for failure.
- Things migrate to available Auths using secure migration protocol.

- When configuring migration policies, Auths consider various constraints including migration costs, feasibility, and communication requirements.

- Auth scans dynamically add or remove available resources including edge computers and things.
- Long-term effects of migration are considered.
- The edge is aware of lifecycles of devices and can predict future resource requirements.

FIGURE 3. The levels of awareness and resilience in the SST.

and authorization requests to Auths. They will notice communication failures with their Auths as events. Event awareness provides the base for higher-level awareness for taking further security measures.

Situation awareness. Situation awareness is used to trigger secure migration. With more resources and better local context awareness than things, Auths can use resource-demanding but more accurate methods. Examples include sharing heartbeat response information for a possibly failed Auth and actively monitoring communication channels to make sure the failure is not attributable to issues in the communication media. Things keep track of failures in the Auth's responses by using simple counters and checking whether the counter value exceeds a certain threshold. By cross-checking the information gathered by Auths and things, the SST infrastructure can determine whether the events indicate a false alarm or an actual failure. When an actual failure is detected, the secure migration process begins.

ADAPTABILITY AWARENESS IS CRITICAL FOR RECOVERING AVAILABILITY IN THE CASE OF DoS ATTACKS.

Adaptability awareness. Auths are aware of the IoT systems' adaptability before failures. Adaptability awareness is critical for recovering availability in the case of DoS attacks. In the SST, Auths construct *migration policies* describing which things should migrate to which Auths when there is an Auth failure. An adaptability-aware migration policy considers factors that affect availability after migration, for example, access requirements between things and trust between Auths.

During normal operations, an Auth sets up *migration credentials*, cryptographic tokens used to establish new trust relationships, for its things and sends them out to other trusted Auths. The Auth also sends a list of trusted Auths and their network addresses to its things. When an Auth failure occurs, the things try sending migration requests to other available trusted Auths. Trusted Auths will accept the migration request when a thing sends a request to the designated Auth or will reject the request otherwise. This scheme allows dynamic changes in migration policies.

Goal awareness. For a given IoT system based on the SST, there can be multiple possible migration policies due to the many combinations of Auths and things. Goal awareness is used to decide which migration policies lead to better availability by considering various constraints including communication costs, the capacity of Auths, load balancing, and signal reachability between things and Auths. Specifically, the SST uses integer linear programming (ILP) to find the best migration policies under given constraints, including the computing power of the edge computer solving the ILP problem. The SST currently supports up to this level of awareness.

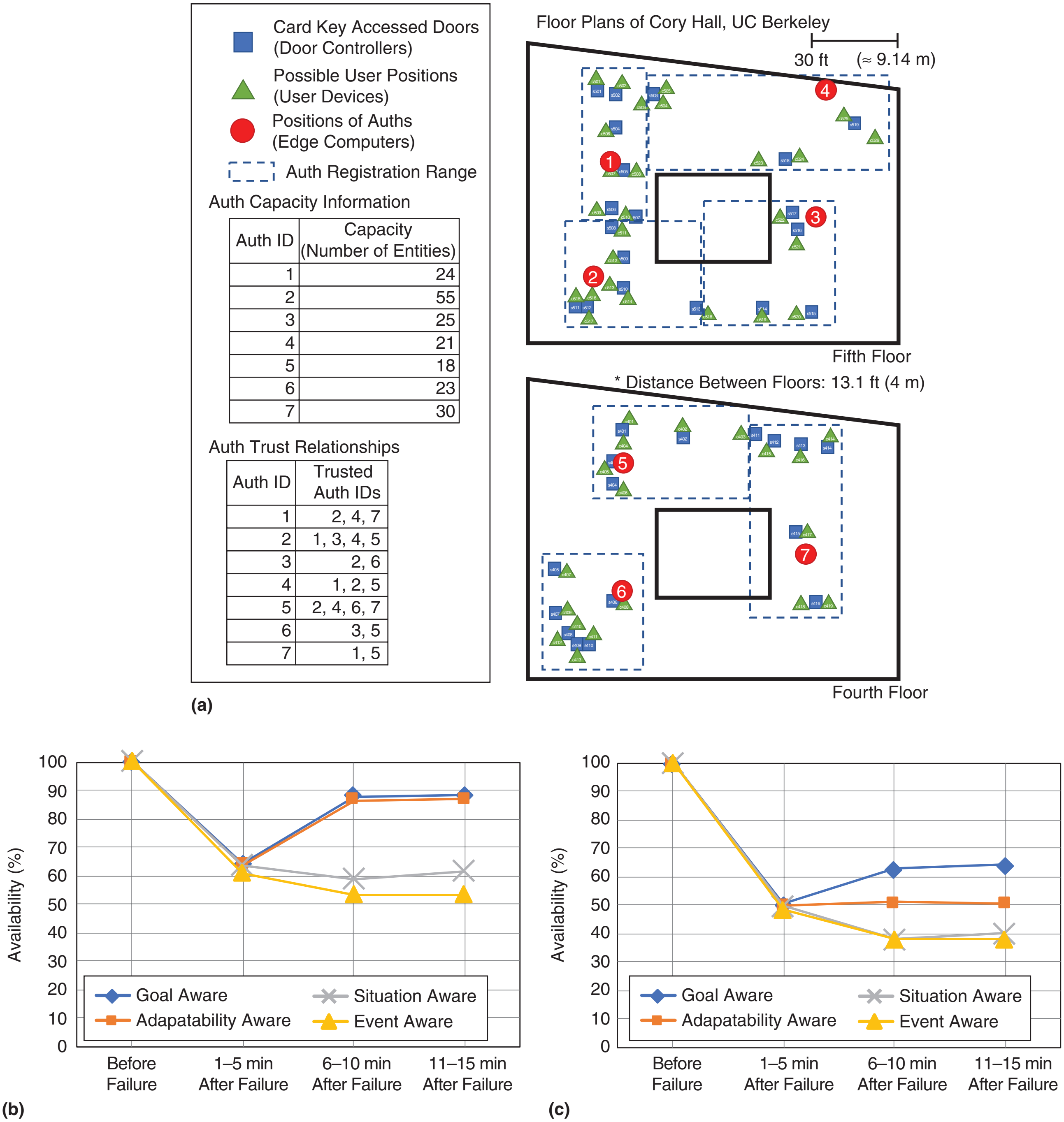
Future awareness. Self-sustainable and future-aware IoT systems should be able to replace and renew worn-out resources. The SST's secure migration can be easily extended to remove or add edge computers hosting Auths. To remove an old Auth, we can set a migration policy without the old Auth, turn it off, and trigger secure migration. When we have a new Auth, we may need to move things from other Auths to the new one for better load balancing. For this, we first set up a migration policy that migrates things to the new Auth, provide the things with new Auth's network information, and enforce migration.

Thanks to the SST's locally centralized and globally distributed architecture, adding and removing things can be done completely locally without any global-level changes. A newly added things will be able to communicate with other devices authorized by other Auths, as long as their Auths maintain trust and allow communication among those things. A removed thing will no longer be able to communicate with others because its Auth will revoke its access.

Auths can formulate and solve ILP problems considering longer-term effects of the short-term migration activities, for example, what if an Auth to which things have migrated fails later? Although the SST's current design is not aware of future resources, we can extend the ILP formulation to include the edge computers expected to be added in the future.

EXPERIMENTS AND RESULTS

To show the resilience of the proposed approach with different context-awareness levels, we carried out experiments by extending the setup



Auth ID	Capacity (Number of Entities)
1	24
2	55
3	25
4	21
5	18
6	23
7	30

Auth ID	Trusted Auth IDs
1	2, 4, 7
2	1, 3, 4, 5
3	2, 6
4	1, 2, 5
5	2, 4, 6, 7
6	3, 5
7	1, 5

FIGURE 4. The experimental setup and results. (a) The experimental setup includes a simulated environment with Auths, door controllers, and user devices with door-opening apps. (b) The graph shows the availability after three Auths (4, 6, and 1) fail. (c) The graph shows the availability after four Auths (4, 6, 1, and 7) fail. UC: University of California.

used in our previous work.[11] Figure 4(a) illustrates the extended experimental setup, a simulated environment of a smart building with door controllers and user devices with door-opening apps. This environment was modeled using floor plans of the fourth and fifth floors of Cory Hall at the University of California, Berkeley, and included seven Auths hosted on edge computers, 35 door controllers, and 45 user devices positioned as shown in Figure 4(a). In this environment, a user device must be authorized by its Auth to open a door. Each user device tried to open the nearby door every minute. Availability was measured by the portion of user devices that successfully opened nearby doors. Each of the Auths, door controllers, and user devices was executed on a Linux container. The network was simulated using the ns-3 simulator (https://www.nsnam.org/) with a wired network for Auth-to-Auth communication and a wireless network for Auth-to-thing and thing-to-thing communication. Each simulation was performed on Amazon Web Services for 20 min in real time, 5 min before Auth failures, and 15 min after failures.

We compared four different awareness levels: event, situation, adaptability, and goal. The event-aware SST was set just to retry and wait for the recovery of the Auths. The situation-aware SST was able to detect Auth failures and trigger an ad hoc migration, which migrates things to nearest Auths first. The migration policy of the adaptability-aware SST considered trust between Auths and communication requirements between things, in this case, which user device should be able to communicate with which door controller. The goal-aware SST also considered the overall system's goal, including the Auths' capacity and load balancing.

The experimental results in Figure 4 show the availability of the experimental IoT system when three [in (b)] and four [in (c)] Auths failed. The results show that higher awareness levels recovered higher availability. For example, the situation-aware SST detected the failures and triggered ad hoc secure migration, but the event-aware SST did not. The adaptability-aware SST could recover even higher availability by considering which Auths could be trusted by things after migration and which things needed to communicate. The goal-aware SST performed better, especially in the case of four Auths failing, because load balancing became more critical when fewer Auths were left after failures.

Guaranteeing availability is critical to making the IoT secure and safe. In addition to the architectural merits of edge computing for countering availability threats, better context awareness leads to a more resilient design of IoT systems, as shown with our authentication and authorization infrastructure for the IoT. The proposed context awareness levels can be concrete guidelines for IoT system designers. Implementing each level of awareness may not always be possible due to constraints; however, it is important to consider lower-level awareness as a foundation on which higher-level awareness is implemented.

As future work, we plan to study awareness levels for other aspects of protecting the IoT. Context awareness can be used to authenticate a user's identity, for instance, based on a user's location or temporal behavior. For event awareness, the edge computers can use sensors to detect other types of DoS attacks, including signal jamming or power drain. Auths can use situation awareness based on statistics to detect application-layer threats, such as service abuse or cybercrimes. Adaptability-aware edge computers can protect the privacy of sensitive information depending on the ongoing agenda in smart conference rooms. Future awareness is the most underexplored area, where we can further research an IoT system's lifecycle speculations, such as demand prediction for IoT services. C

ACKNOWLEDGMENTS

The work in this article was supported in part by the National Science Foundation award CNS-1836601 (Reconciling Safety With the Internet for Cyber-Physical) and iCyPhy (the Industrial Cyber-Physical Systems Research Center), which is supported by Avast, Denso, Ford, and Siemens.

REFERENCES

1. L. Mathews, "Hackers use DDoS attack to cut heat to apartments," *Forbes*, Nov. 2016. [Online]. Available: https://www.forbes.com/sites/leemathews/2016/11/07/ddos-attack-leaves-finnish-apartments-without-heat/
2. Corero Network Security, "Corero DDoS trends report—Q2-Q3 2017," Marlborough, MA, 2017. [Online]. Available: http://info.corero.com/rs/258-JCF-941/images/2017-q2q3-ddos-trends-report.pdf
3. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017. doi: 10.1109/MC.2017.201.

4. I. Morris, “Google’s latest failure shows how immature its hardware is,” *Forbes*, Feb. 2017. [Online]. Available: http://www.forbes.com/sites/ianmorris/2017/02/24/googles-latest-failure-shows-how-immature-its-hardware-is/
5. W. Shi and S. Dustdar, “The promise of edge computing,” *Computer*, vol. 49, no. 5, pp. 78–81, May 2016. doi: 10.1109/MC.2016.145.
6. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proc. 1st MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16.
7. A. Botta, W. de Donato, V. Persico, and A. Pescap, “Integration of cloud computing and Internet of Things: A survey,” *Future Generation Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016. doi: 10.1016/j.future.2015.09.021.
8. H. Kim, E. Kang, E. A. Lee, and D. Broman, “A toolkit for construction of authorization service infrastructure for the Internet of Things,” in *Proc. 2nd ACM/IEEE Int. Conf. Internet-of-Things Design and Implementation*, Pittsburgh, PA, Apr. 2017, pp. 147–158.
9. H. Kim and E. A. Lee, “Authentication and authorization for the Internet of Things,” *IT Professional*, vol. 19, no. 5, pp. 27–33, Oct. 2017. doi: 10.1109/MITP.2017.3680960.
10. H. Kim, A. Wasicek, B. Mehne, and E. A. Lee, “A secure network architecture for the Internet of Things based on local authorization entities,” in *Proc. 4th IEEE Int. Conf. Future Internet of Things and Cloud*, Vienna, Austria, Aug. 2016, pp. 114–122.
11. H. Kim, E. Kang, D. Broman, and E. A. Lee, “An architectural mechanism for resilient IoT services,” in *Proc. 1st ACM Workshop on Internet of Safe Things*, Delft, The Netherlands, Nov. 2017, pp. 8–13.
12. M. Lohstroh, H. Kim, and E. A. Lee, “Contextual callbacks for resource discovery and trust negotiation on the Internet of Things: work-in-progress,” in *Proc. 13th ACM Int. Conf. Embedded Software*, Seoul, South Korea, Oct. 2017, pp. 14:1–14:2.
13. Y. Ben Saied, A. Olivereau, D. Zeghlache, and M. Laurent, “Trust management system design for the Internet of Things: A context-aware and multi-service approach,” *Comput. Security*, vol. 39, pp. 351–365, Nov. 2013. doi: 10.1016/j.cose.2013.09.001.

ABOUT THE AUTHORS

HOKEUN KIM was with the University of California, Berkeley and is now a software engineer with Google, where he works on Internet security research on the Safe Browsing team. His research interests include computer security, the Internet of Things, and cyberphysical systems. Kim received a Ph.D. in electrical engineering and computer sciences from the University of California, Berkeley, in 2017. He received an ACM/IEEE Best Paper Award and an IEEE Honorable Mention in 2017. Contact him at hokeunkim@eecs.berkeley.edu.

EDWARD A. LEE is the Robert S. Pepper Distinguished Professor in the Graduate School in Electrical Engineering and Computer Sciences at the University of California, Berkeley, and director of iCyPhy, the Berkeley Industrial Cyber-Physical Systems Research Center, and of the Berkeley Ptolemy project. His research interests include cyberphysical systems, which integrate physical dynamics with software and networks, with a focus on the use of deterministic models as a central part of the engineering toolkit for such systems. Lee received a Ph.D. in electrical engineering and computer sciences from the University of California, Berkeley, in 1986. He was an NSF Presidential Young Investigator and received the 1997 Frederick Emmons Terman Award for Engineering Education, the 2016 Outstanding Technical Achievement and Leadership Award from the IEEE Technical Committee on Real-Time Systems, and the 2018 Berkeley Citation. He is a Fellow of the IEEE. Contact him at eal@eecs.berkeley.edu.

SCHAHRAM DUSTDAR is a full professor of computer science heading the Distributed Systems Research Division at Technische Universität Wien, Austria. His research interests include the Internet of Things and edge computing. Dustdar received a Habilitation degree in computer science from Technische Universität Wien. He is an IEEE Fellow, an Association for Computing Machinery Distinguished Scientist, a member of the Academia Europaea—The Academy of Europe, and a member of the *Computer* Editorial Board. Contact him at dustdar@dsg.tuwien.ac.at.