EdgeCons: Achieving Efficient Consensus in Edge Computing Networks

Zijiang Hao Shanhe Yi Qun Li
College of William & Mary

Abstract

Fast event ordering is critical for delay-sensitive edge computing applications that serve massive geographically distributed clients. Using a centralized cloud to determine the event order suffers from unsatisfactory latency. Naive edge-centric solutions, which designate one edge node to order all the events, have scalability and single point of failure issues. To address these problems, we propose EdgeCons, a novel consensus algorithm optimized for edge computing networks. EdgeCons achieves fast consensus by running a sequence of Paxos instances among the edge nodes and dynamically distributing their leadership based on the recent running history. It also guarantees progressiveness by incorporating a reliable, backend cloud. A preliminary evaluation shows that EdgeCons works more efficiently than the state-of-the-art consensus algorithms, in the context of achieving fast event ordering in edge computing networks.

1 Introduction

Edge computing (aka. fog computing [4], cloudlet [33] and MEC [31]) is a concept that has gained lots of attention recently [32, 35, 43]. The basic idea of edge computing is to provide elastic resources like cloud computing at the edge of network, such that the requests for the resources from client devices will be handled by edge computing nodes at places closer to the users. Modern applications and services, such as real-time video processing, cognitive analytics in critical missions and online video games, can benefit a lot from the decentralized edge computing paradigm. Clients connected to the edge will experience faster service response. The nearby access saves network bandwidth and lowers the peak workload to the cloud. The deployment of applications can adapt to the geographic distribution of users and make the most efficient usage of the resources on the edge nodes. As edge computing possesses so many advantages, we expect more and more applications and services be deployed on edge computing platforms.

When massive client devices are connected to the edge, a large volume of events will be generated at the edge that need to be synchronized in a consistent order among all the edge nodes. Event ordering is fundamental for the correctness of distributed systems [17]. A cloud-centric solution employs cloud servers to order all the events. The drawback is that it may take a long time to determine the order of the events, as massive events have to go to the cloud in the first place. Because many modern applications and services are human-centric, latency has a huge impact on the user experience. For example, lower latency will greatly improve the user experience of games which highly rely on the fast reaction of the players [6]. Therefore, in this paper, we study how to achieve fast event ordering for large-scale delay-sensitive distributed applications in edge computing networks.

It is quite challenging to achieve this goal. In contrast to cloud networks and the cloud servers, edge computing networks are usually heterogeneous, and the edge computing nodes are less reliable. Events transmitted over such a network are subject to unpredictable delivery time and packet loss rate, not to mention the large amount of events received by all the edge nodes for ordering. To be noted that, while edge nodes are less reliable, edge node failures are still rare. Therefore, opportunistically using a local edge computing network to order the events has performance gain compared to barely using a remote cloud. In our previous study [41], we have shown that the Amazon EC2 cloud leads to much higher latency than a locally-built edge computing network. A naive edge-centric solution can simply select an edge node to order all the events. The drawback is that the designated edge node will become the performance bottleneck of the system and a single point of failure. Using timestamps to order the events is also a bad idea, which has been thoroughly discussed in the literature [29, 18].

To the best of our knowledge, we are the first to study the consensus approaches for edge computing networks. Most existing researches of edge computing focus on the client-to-edge interactions, such as how to accommodate computation and storage tasks for the clients on edge nodes. Our problem concentrates on event ordering in the scope of interconnected edge nodes, and we will show the traits of this problem via an online gaming application. Existing consensus protocols can hardly meet the latency requirements imposed by this problem. Multi-Paxos [19] runs Paxos instances with a designated

leader. Since edge nodes are not as reliable and powerful as cloud servers, the leader edge node will encounter the same issues as in the naive edge-centric solution. Mencius [24] addresses the issues by assigning the leadership evenly among the system nodes. The main problem of Mencius is that a slow system node will deteriorate the overall system performance. E-Paxos [25] avoids this problem by opportunistically executing Fast Paxos [21] instances instead of Paxos instances. Nevertheless, it suffers from heavier network burdens than Multi-Paxos and Mencius for achieving consensus.

To this end, we design EdgeCons, a consensus algorithm that achieves fast event ordering for large-scale distributed applications in edge computing networks. Similar to Mencius, EdgeCons runs a sequence of Paxos instances on the edge, but it distributes the leadership of the Paxos instances based on the recently running history of the consensus process. EdgeCons also guarantees the progressiveness of consensus, which is indispensable for the edge computing scenario. It achieves so by employing a backend cloud that never fails or becomes network-partitioned as a reliable conflict resolver in the system.

To summarize, we make the following contributions.

- We formulate the problem of achieving fast event ordering for large-scale delay-sensitive distributed applications in edge computing networks, and propose realistic application scenarios related to this problem.
- We design a novel consensus solution to this problem, which dynamically distributes the leadership of a Paxos instance sequence among the edge nodes based on the recent running history, and guarantees progressiveness by incorporating a reliable, backend cloud.
- Preliminary simulation results reveal that our solution works more efficiently than existing consensus solutions in the literature, in the context of achieving fast event ordering in edge computing networks.

2 Preliminaries

Before digging into our solution, we briefly introduce some preliminaries on edge computing and consensus.

Edge Computing. Edge computing aims at serving the end users at the edge of network, providing better network conditions than cloud computing. It has attracted a lot of research effort recently [2, 14, 26, 36, 28, 1, 15, 10, 7, 34, 40, 22, 44, 12, 11, 42, 23]. Figure 1 depicts a typical architecture of edge computing. Client devices, such as wearables, smartphones, tablets and laptops, are connected via wireless links to the level 1 edge nodes, which are mostly wireless access points and cellular base stations with extra hardware resources. Level 1 edge nodes are backed by level 2 edge nodes, which possess more powerful hardware resources but have longer network distances to the client devices. Level 2 edge nodes

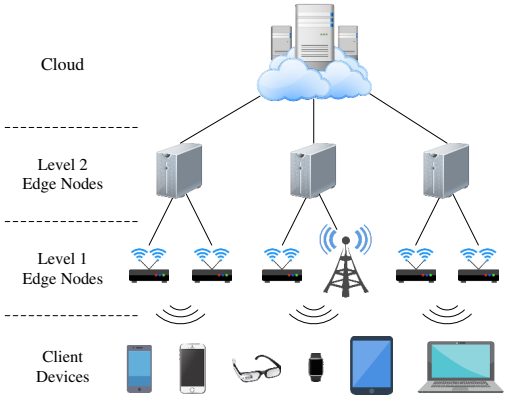

Figure 1: A typical architecture of edge computing.

are further backed by a cloud at the core of network. It is worth mentioning that Figure 1 only showcases one typical architecture of edge computing among many others. Other edge computing architectures may contain less than or more than two levels of edge nodes. In addition, some architectures may have no cloud at the backend.

Consensus. Consensus is a fundamental problem in distributed computing: How to achieve overall system reliability in the presence of a number of faulty nodes? It has been studied for decades but remains a hot research topic in academia [18, 19, 21, 20, 3, 24, 25, 30, 16, 46, 8, 45, 38]. Among existing consensus solutions, we focus on the Paxos-based ones. On one hand, to the best of our knowledge, most, if not all, consensus solutions that achieve strong consistency under the typical FLP [9] setting are essentially variants of Paxos, such as Raft [30]. On the other hand, other consensus solutions, such as the blockchain-based ones [27, 39], cannot guarantee strong consistency, thus violating our design goal. Note that an execution of the Paxos algorithm is usually called a Paxos "instance" in the literature. There are two types of phases in a classical Paxos instance: Phase 1 is for electing the leader, while Phase 2 is for deciding a value on behalf of the entire system.

As the Paxos-based solutions employ the so-called "state machine replication" method, consensus is also related to the research problem of how to build replicated and reliable services for the users across a large geographic area. Among the literature on consensus, three Paxos-based consensus solutions, i.e., Multi-Paxos [19], Mencius [24] and E-Paxos [25], are more related to our solution than the others. Multi-Paxos runs a sequence of Paxos instances with a designated leader. Mencius improves Multi-Paxos by distributing the leadership evenly among the nodes. E-Paxos opportunistically runs Fast Paxos [21] instances instead of Paxos instances, and improves the system performance by delaying the resolution of conflicts. EdgeCons shares some similarities with

Mencius. Nevertheless, it is different from all the aforementioned consensus solutions, because it is optimized for the application scenarios found in edge computing environments. In-depth discussion will be provided in the following part of the paper.

3 Deriving EdgeCons

In this section, we first propose an application scenario for which EdgeCons is designed, and then discuss the approach of EdgeCons in detail.

3.1 A Motivational Scenario

To achieve a better understanding on the problem EdgeCons tries to solve, consider the following scenario. A game company operates a massive multi-player online gaming service to the players located in a city. The players can connect their own devices, such as smartphones and AR devices, to a remote game server through wireless links. The latency perceived by the players is critical in achieving satisfactory gaming experience, and a large number of players may play the game simultaneously. As such, it is not a good choice to perform the main part of the computation for each player on the remote server, no matter it is a cloud server or an edge node. This is because a cloud server cannot provide low enough latency [41, 13], while an edge node cannot afford the heavy computation for so many players. We hence make the following assumptions for this scenario.

- The main part of computation for each player is done locally on the player's client device, and the main task of the remote gaming service is to order the input from all the online players.
  More specifically, the client device runs a piece of gaming software that forwards the player's input to the remote gaming service and receives the ordered input by all the online players. The gaming software also performs computation on the ordered input and renders the result to the player. The computation performed on the client device is deterministic, similar to what has been discussed in the literature [8], such that all the players have consistent views of the game.
- The input forwarded to the remote gaming service is of a small size. It may include the data collected from the touch screen, the accelerator, and/or the GPS equipped on the client device, but does not contain any multimedia data such as the voice of the player. The output of the remote gaming service is thus also of a small size, compared to the video stream output typically seen by existing remote gaming solutions. We also assume that the client devices are powerful enough to perform the computation of the game locally [37].

As such, the game company cannot adopt user-side solutions, as long as it wants to guarantee strong consistency on the event order across the system. The best design choice that the game company can make, from our point of view, is to build and utilize an edge computing network for input ordering. To be more specific, the company sets or rents several edge nodes in the city, and deploys the input-ordering service on them. The edge nodes are well scattered, such that their service regions cover the whole city with modest overlapping, and they can communicate with each other through a wide area network (WAN). The players connect their client devices to the nearest edge nodes, interacting with the input-ordering service. As discussed in Section 1, naive solutions cannot work properly or efficiently in such a situation, while existing consensus algorithms likewise cannot fully exploit the potential of the edge computing network. This motivates us to design EdgeCons, a novel consensus algorithm for achieving fast event ordering in edge computing networks.

3.2 Approach

As mentioned previously, EdgeCons shares some similarities with Mencius. In particular, EdgeCons distributes the leadership of the Paxos instances among the edge nodes, and when an edge node works as the leader, it starts from the state in which it has already run Phase 1 for the initial round. Therefore, all the leader edge nodes skip Phase 1 and start from Phase 2, which improves the system performance.

Nevertheless, EdgeCons is different from Mencius in the following aspects. First, EdgeCons does not distribute the leadership evenly like Mencius, but based on the running history of the system. This allows EdgeCons to distribute the leadership more wisely, and makes the system performance better. Second, unlike Mencius, EdgeCons relaxes the assumption that the network links between the edge nodes are FIFO. The edge nodes are free to use non-FIFO links, such as links based on UDP, and the messages transmitted via those links may be reordered during the transmission. Last, EdgeCons assumes that there is a cloud behind the edge network that never fails or becomes network-partitioned. By incorporating such a backend cloud into the system, EdgeCons breaks the assumption made by the FLP paper [9], and guarantees the progressiveness of the consensus process.

We summarize the rules of EdgeCons as follows.

- EdgeCons divides the consensus process into epochs. In each epoch, EdgeCons executes a sequence of Paxos instances, with their leadership pre-distributed among the edge nodes. The number of Paxos instances in each epoch is predefined and fixed, denoted by N_{Paxos}. Similar to Mencius, the Paxos instances are

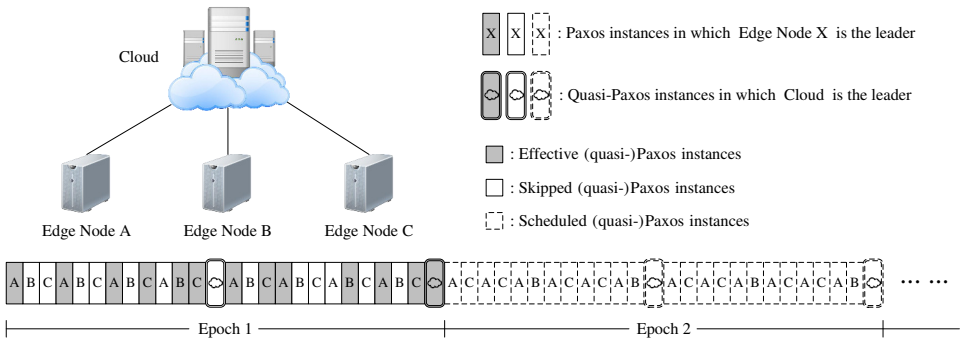

Figure 2: The leadership distribution method of EdgeCons. The edge computing system in the example consists of three edge nodes (A, B and C) and a backend cloud. The first two epochs are shown in the example.

executed from Phase 2; Phase 1 is considered already executed by the leader for the initial round.

- Unlike Mencius, which distributes the leadership of the Paxos instances evenly among all the nodes, EdgeCons distributes the leadership dynamically according to the running history of the system. More specifically, EdgeCons counts the number of the effective Paxos instances under the leadership of each edge node in the previous N_{epoch} epochs (N_{epoch} is a predefined positive value), denoted by $N_{ef,i}$ for edge node i ($i = 1, .., n$), and distributes the leadership for the next epoch accordingly, proportional to the $N_{ef,i}$ values. Note that a Paxos instance is considered effective if and only if it has made the system agree on a proposed value.
- To distribute the leadership for the next epoch, EdgeCons arranges the upcoming Paxos instances in a way that the Paxos instances are assigned to each edge node as evenly-scattered in the epoch as possible, such that they are separated by the same or similar numbers of Paxos instances assigned to the others. Because of the deterministicity of this algorithm, the edge nodes can learn the distribution of leadership for the upcoming epoch independently, without the need of interacting with the other edge nodes.
- EdgeCons also involves a backend cloud into the algorithm. The cloud resides at the core of network, and is therefore less efficient than the edge nodes in exchanging the messages of events. Nevertheless, it is far more reliable than the edge nodes. We assume that the cloud never fails or becomes network-partitioned, so it is always reachable to the edge nodes.
- In addition to the Paxos instances assigned to the edge nodes, there are also quasi-Paxos instances assigned to the cloud in each epoch, which work as follows. When an edge node has failed to make the system agree on a value it intends to propose for N_{fail} times (N_{fail} is a pre-defined positive value), it transmits the value to the cloud. The cloud collects all such values, orders them by their arrival time, and announces them to the edge nodes in the next quasi-Paxos instance.
- When a quasi-Paxos instance is over, the cloud will initiate the next quasi-Paxos instance by sending the values it has collected in-between the two quasi-Paxos instances to all the edge nodes. In case that no value has been collected, the cloud will send a SKIP message to the edge nodes. The edge nodes must accept the values (or the SKIP message), and reply an OKAY message to the cloud. Having collected the OKAY messages from a majority of the edge nodes, the cloud considers that the current quasi-Paxos instance is over, and starts the next quasi-Paxos instance.
- The edge nodes cannot skip the quasi-Paxos instances. In other words, they cannot execute the Paxos instances posterior to each of the quasi-Paxos instances they have encountered, until they have received either the ordered values or a SKIP message from the cloud.

The consensus instances assigned to the cloud are called quasi-Paxos instances, because they are similar to at first glance, but fundamentally different from the Paxos instances in Multi-Paxos. Being highly reliable, the cloud can actually guarantee consensus without collecting the OKAY messages from a majority of the edge nodes. We design EdgeCons in a way that the cloud collects the OKAY messages only for the performance considerations. Moreover, the cloud is not included in any quorum, no matter the consensus instance is a Paxos one or a quasi-Paxos one. The main purpose of involving the backend cloud, as it can be seen, is to help the edge nodes propose values when their Paxos instances have been frequently skipped by the others for whatever reasons. This guarantees the progressiveness of the consensus process, and sets a logical "upper bound" on the latency perceived by the clients. Note that this statement does not violate the FLP result [9], because EdgeCons makes a different assumption by involving the cloud. The cloud is barely an arbitrator, but not a replica as the edge nodes, because

it does not maintain a local state for the event log. Finally, in EdgeCons, the leader edge node proposes at most one value in a Paxos instance, but the cloud can propose potentially many values in a quasi-Paxos instance. Since the values are of small data amount, proposing many values does not make a big difference in the transmission time with proposing one value.

It is also worth mentioning that the quasi-Paxos instances should be well distributed in each epoch, such that the cloud executes them continuously, and the edge nodes cross them naturally without being blocked for a long time. In addition, EdgeCons makes a strong assumption that the cloud never fails or becomes network-partitioned. In fact, clouds in the real world sometimes do experience temporary outages [5]. However, we believe that it is still possible to build a highly reliable cloud-based arbitrator in practice, e.g., by employing the clouds from different companies (Google, Amazon, Microsoft, etc.) and utilizing Paxos to coordinate them.

Figure 2 depicts a simple example of how EdgeCons works. Due to the space limit, the figure only shows the first two epochs of the consensus process, and each epoch only contains 24 Paxos instances and 2 quasi-Paxos instances. In spite of its simplicity, Figure 2 depicts some important aspects of EdgeCons. In Epoch 1, for example, the leadership of the Paxos instances is distributed in a round-robin way as Mencius. When Epoch 1 ends, Edge Node A has contributed 6 effective Paxos instances, Edge Node B has contributed 2 and Edge Node C has contributed 4. Consequently, in the upcoming Epoch 2, Edge Node A possesses half of the Paxos instances, Edge Node B possesses one sixth and Edge Node C possesses one third. Since Epoch 1 is the only epoch before Epoch 2, EdgeCons can only refer to this epoch when determining the leadership distribution in Epoch 2. For the following epochs, however, more epochs can be referred to as long as N_{epoch} is larger than 1.

4 Evaluation

To examine the efficiency of EdgeCons, we conduct a first-step simulation experiment with the following settings. The round-trip time (RTT) between the client and the edge is 10 ms, and the edge nodes have a RTT of 10 ms to the others, except a slow one, which has a RTT of 40 ms to the others. The cloud has a RTT of 60 ms to the edge. Each edge node can only transmit no more than 10,000 messages to the others in one second. The workload is that 2,000 events have been sent from the client to each edge node in one second, with their intervals uniformly distributed. Two cases, i.e., a system with 5 edge nodes and another with 7 edge nodes, are tested. The edge nodes in EdgeCons will ask the cloud to propose a value after two failures, those in Mencius will try to skip a Paxos instance after waiting for 80 ms, and those in E-Paxos will try to urge the execution of a consensus instance after waiting for 80 ms. With these settings, we compare the user-perceived latency under different consensus algorithms. Figure 3 depicts the results.

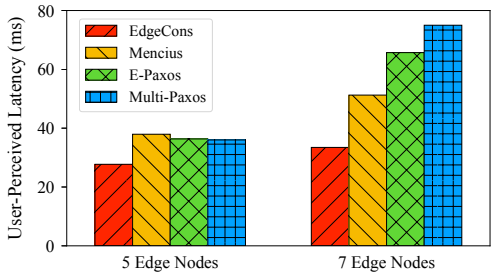

Figure 3: A comparison of the user-perceived latency.

It can be seen from Figure 3 that EdgeCons achieves the best performance in both cases. Note that the edge network is assumed non-FIFO: Mencius cannot piggyback SKIP messages on other messages. Consequently, Mencius sends more messages than EdgeCons, especially when there is a slow edge node in the system. E-Paxos also needs to deal with more messages, and it is likely to encounter more conflicts than EdgeCons. Multi-Paxos suffers from the problem that the leader is the performance bottleneck. It is also worth mentioning that in the second case of the experiment, Mencius, E-Paxos and Multi-Paxos are overwhelmed by the workload, which lasts for one second in the simulation. With the increase of the workload duration, the user-perceived latency of these three algorithms will increase dramatically.

5 Conclusion & Future Work

In this paper, we propose a novel consensus protocol, EdgeCons, for achieving fast event ordering in edge computing networks. A preliminary evaluation reveals that EdgeCons works better than the state-of-the-art consensus algorithms.

We will improve EdgeCons in the following two directions in our future work. First, the current leadership distribution method has a drawback. If an edge node has experienced a network congestion but later recovers, it may take a long time to re-gain its “leadership share”. For this reason, a deterministic, randomized method that can tune the leadership share in a timely manner should be designed in the future. Second, how to monitor the system effectively, such that EdgeCons can quickly adjust the number of the quasi-Paxos instances in the upcoming epochs, in response to the dramatic changes on the workload received by the edge, is a challenging task and requires more considerations.

References

[1] ANGLANO, C. F., CANONICO, M., CASTAGNO, P., GUAZZONE, M., AND SERENO, M. A game-theoretic approach to coalition formation in fog provider federations. In *Proceedings of the Third IEEE International Conference on Fog and Mobile Edge Computing* (2018), FMEC '18, pp. 2–3.

[2] BHARDWAJ, K., SHIH, M.-W., AGARWAL, P., GAVRILOVSKA, A., KIM, T., AND SCHWAN, K. Fast, scalable and secure onloading of edge functions using airbox. In *Proceedings of 2016 IEEE/ACM Symposium on Edge Computing* (2016), SEC '16, pp. 14–27.

[3] BIELY, M., MILOSEVIC, Z., SANTOS, N., AND SCHIPER, A. S-paxos: Offloading the leader for high throughput state machine replication. In *Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems* (2012), SRDS '12, pp. 111–120.

[4] BONOMI, F., MILITO, R., ZHU, J., AND ADDEPALLI, S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing* (2012), MCC '12, pp. 13–16.

[5] BORT, J. Google apologizes for cloud outage. `http://www.businessinsider.com/google-apologizes-for-cloud-outage-2016-4`, 2016.

[6] BURGER, V., PAJO, J. F., SANCHEZ, O. R., SEUFERT, M., SCHWARTZ, C., WAMSER, F., DAVOLI, F., AND TRAN-GIA, P. Load dynamics of a multiplayer online battle arena and simulative assessment of edge server placements. In *Proceedings of the 7th International Conference on Multimedia Systems* (2016), MMSys '16, pp. 17:1–17:9.

[7] CHEN, Z., HU, W., WANG, J., ZHAO, S., AMOS, B., WU, G., HA, K., ELGAZZAR, K., PILLAI, P., KLATZKY, R., SIEWIOREK, D., AND SATYANARAYANAN, M. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 14:1–14:14.

[8] CUI, H., GU, R., LIU, C., CHEN, T., AND YANG, J. Paxos made transparent. In *Proceedings of the 25th Symposium on Operating Systems Principles* (2015), SOSP '15, pp. 105–120.

[9] FISCHER, M. J., LYNCH, N. A., AND PATERSON, M. S. Impossibility of distributed consensus with one faulty process. *J. ACM 32*, 2 (1985), 374–382.

[10] HA, K., ABE, Y., EISZLER, T., CHEN, Z., HU, W., AMOS, B., UPADHYAYA, R., PILLAI, P., AND SATYANARAYANAN, M. You can teach elephants to dance: Agile vm handoff for edge computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 12:1–12:14.

[11] HAO, Z., AND LI, Q. Edgestore: Integrating edge computing into cloud-based storage systems: Poster abstract. In *Proceedings of 2016 IEEE/ACM Symposium on Edge Computing* (2016), SEC '16, pp. 115–116.

[12] HAO, Z., NOVAK, E., YI, S., AND LI, Q. Challenges and software architecture for fog computing. *IEEE Internet Computing 21*, 2 (2017), 44–53.

[13] HAO, Z., TANG, Y., ZHANG, Y., NOVAK, E., CARTER, N., AND LI, Q. Smoc: A secure mobile cloud computing platform. In *Proceedings of 2015 IEEE International Conference on Computer Communications* (2015), INFOCOM '15, pp. 2668–2676.

[14] JANG, M., LEE, H., SCHWAN, K., AND BHARDWAJ, K. Soul: An edge-cloud system for mobile applications in a sensor-rich world. In *Proceedings of 2016 IEEE/ACM Symposium on Edge Computing* (2016), SEC '16, pp. 155–167.

[15] JIANG, Y., HUANG, Z., AND TSANG, D. H. K. Challenges and solutions in fog computing orchestration. *IEEE Network PP*, 99 (2017), 1–8.

[16] KRASKA, T., PANG, G., FRANKLIN, M. J., MADDEN, S., AND FEKETE, A. Mdcc: Multi-data center consistency. In *Proceedings of the 8th ACM European Conference on Computer Systems* (2013), EuroSys '13, pp. 113–126.

[17] LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM 21*, 7 (1978), 558–565.

[18] LAMPORT, L. The part-time parliament. *ACM Transactions on Computer Systems 16*, 2 (1998), 133–169.

[19] LAMPORT, L. Paxos made simple. *ACM SIGACT News 32*, 4 (2001), 18–25.

[20] LAMPORT, L. Generalized consensus and paxos. Tech. rep., MSR-TR-2005-33, Microsoft Research, 2005.

[21] LAMPORT, L. Fast paxos. *Distributed Computing 19*, 2 (2006), 79–103.

[22] LEE, K., FLINN, J., AND NOBLE, B. D. Gremlin: Scheduling interactions in vehicular computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 4:1–4:13.

[23] MA, L., YI, S., AND LI, Q. Efficient service handoff across edge servers via docker container migration. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 11:1–11:13.

[24] MAO, Y., JUNQUEIRA, F. P., AND MARZULLO, K. Mencius: Building efficient replicated state machines for wans. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (2008), OSDI '08, pp. 369–384.

[25] MORARU, I., ANDERSEN, D. G., AND KAMINSKY, M. There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (2013), SOSP '13, pp. 358–372.

[26] MORTAZAVI, S. H., SALEHE, M., GOMES, C. S., PHILLIPS, C., AND DE LARA, E. Cloudpath: A multi-tier cloud computing framework. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 20:1–20:13.

[27] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[28] NASTIC, S., TRUONG, H.-L., AND DUSTDAR, S. A middleware infrastructure for utility-based provisioning of iot cloud systems. In *Proceedings of 2016 IEEE/ACM Symposium on Edge Computing* (2016), SEC '16, pp. 28–40.

[29] OKI, B. M., AND LISKOV, B. H. Viewstamped replication: A new primary copy method to support highly-available distributed systems. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing* (1988), PODC '88, pp. 8–17.

[30] ONGARO, D., AND OUSTERHOUT, J. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (2014), USENIX ATC '14, pp. 305–320.

[31] PATEL, M., NAUGHTON, B., CHAN, C., SPRECHER, N., ABETA, S., NEAL, A., ET AL. Mobile-edge computing introductory technical white paper. *White Paper, Mobile-Edge Computing (MEC) Industry Initiative* (2014).

[32] SATYANARAYANAN, M. The emergence of edge computing. *Computer 50*, 1 (2017), 30–39.

[33] SATYANARAYANAN, M., BAHL, P., CACERES, R., AND DAVIES, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing 8*, 4 (2009), 14–23.

[34] SATYANARAYANAN, M., SCHUSTER, R., EBLING, M., FETTWEIS, G., FLINCK, H., JOSHI, K., AND SABNANI, K. An open ecosystem for mobile-cloud convergence. *IEEE Communications Magazine 53*, 3 (2015), 63–70.

[35] SHI, W., CAO, J., ZHANG, Q., LI, Y., AND XU, L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal 3*, 5 (2016), 637–646.

[36] SHI, W., AND DUSTDAR, S. The promise of edge computing. *Computer 49*, 5 (2016), 78–81.

[37] TRIGGS, R. How far we've come: a look at smartphone performance over the past 7 years. `https://www.androidauthority.com/smartphone-performance-improvements-timeline-626109/`, 2015.

[38] WEI, W., GAO, H. T., XU, F., AND LI, Q. Fast mencius: Mencius with low commit latency. In *Proceedings of 2013 IEEE International Conference on Computer Communications* (2013), INFOCOM '13, pp. 881–889.

[39] WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. `http://gavwood.com/paper.pdf`, 2014.

[40] XIAO, Y., AND ZHU, C. Vehicular fog computing: Vision and challenges. In *Proceedings of 2017 IEEE International Conference on Pervasive Computing and Communications Workshops* (2017), PerCom Workshops '17, pp. 6–9.

[41] YI, S., HAO, Z., QIN, Z., AND LI, Q. Fog computing: Platform and applications. In *Proceedings of 2015 the Third IEEE Workshop on Hot Topics in Web Systems and Technologies* (2015), HotWeb '15, pp. 73–78.

[42] YI, S., HAO, Z., ZHANG, Q., ZHANG, Q., SHI, W., AND LI, Q. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing* (2017), SEC '17, pp. 15:1–15:13.

[43] YI, S., LI, C., AND LI, Q. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data* (2015), MoBiData '15, pp. 37–42.

[44] YI, S., QIN, Z., AND LI, Q. Security and privacy issues of fog computing: A survey. In *Proceedings of the 10th International Conference on Wireless Algorithms, Systems, and Applications* (2015), WASA '15, pp. 685–695.

[45] ZHANG, I., SHARMA, N. K., SZEKERES, A., KRISHNAMURTHY, A., AND PORTS, D. R. K. Building consistent transactions with inconsistent replication. In *Proceedings of the 25th Symposium on Operating Systems Principles* (2015), SOSP '15, pp. 263–278.

[46] ZHAO, M., AND FIGUEIREDO, R. J. Application-tailored cache consistency for wide-area file systems. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems* (2006), ICDCS '06, pp. 41–41.