**Umakishore Ramachandran, Harshit Gupta, Adam Hall, Enrique Saurez and Zhuangdi Xu** *Georgia Institute of Technology, Atlanta, Georgia*

**Editor: Shadi Noghabi**

# A CASE FOR ELEVATING THE EDGE TO BE A PEER OF THE CLOUD

Over the last 20 years, mobile computing has evolved to encompass a wide array of increasingly data-rich applications. Many of these applications were enabled by the Cloud computing revolution, which commoditized server hardware to support vast numbers of mobile users from a few large, centralized data centers. Today, mobile's next stage of evolution is spurred by interest in emerging technologies such as Augmented and Virtual Reality (AR/VR), the Internet of Things (IoT), and Autonomous Vehicles. New applications relying on these technologies often require very low latency response times, increased bandwidth consumption, and the need to preserve privacy. Meeting all of these requirements from the Cloud alone is challenging for several reasons. First, the amount of data generated by devices can quickly saturate the bandwidth of backhaul links to the Cloud. Second, achieving low-latency responses for making decisions on sensed data becomes increasingly difficult the further mobile devices are from centralized Cloud data centers. And finally, regulatory or privacy restrictions on the data generated by devices may require that such data be kept locally. For these reasons, enabling next-generation technologies requires us to reconsider the current trend of serving applications from the Cloud alone.

Recently there has been a growing interest in the use of Edge computing to meet the unique challenges posed by emerging mobile applications. In the Edge computing paradigm, applications are served from a series of micro data centers that are geographically distributed throughout the last mile of the network. These micro data centers may exist in a variety of locations, such as the wiring closet of a hotel or the base of a cell tower. Given their small physical footprint, the amount of hardware resources that they can provide are considerably more limited and far more heterogeneous. Likewise, facilities such as power, cooling, and network connectivity can vary from location to location. Supporting next-generation applications at the Edge requires us to work within these limitations to achieve a high degree of multi-tenancy while providing the Cloud-like experiences to which application developers have become accustomed. By collocating computing resources close to client devices, it becomes feasible to provide several points of data aggregation with low-latency connectivity and localized processing and storage of data. However, realizing this vision is not as simple as moving applications directly from Cloud to Edge. While there has been industry movement towards Edge computing [1, 2] this movement has been incremental and limited to a client-server relationship with the Cloud. We believe that there is a need to rethink the system-level abstractions for hosting the next generation mobile applications in the horizontal and vertical continuum of computational elements in the Edge-Cloud ecosystem (Figure 1). Specifically, we argue the need for elevating the Edge to be a peer of the Cloud.

## A NEXT GENERATION APPLICATION FOR THE EDGE

We can better understand the need for the Edge as a peer to the Cloud by looking at the next evolution of one of mobile's killer apps:

Photo, istockphoto.com

social media. The advent of this technology has enabled people to share photos, videos, and text with friends and family across the world. The next generation of social media applications will be far more immersive. People will use phones and other AR devices to bridge their virtual interactions with the real world. These interactions will be bolstered by the rich streams of data from IoT devices, providing unprecedented experiences.

Consider a scenario where people gather for a parade, such as the annual Macy's Thanksgiving Day Parade in New York City. Events of this type can host millions of spectators and thousands of exhibitors over parade routes spanning many city miles. [3] In our futuristic scenario, each spectator has a camera-equipped AR device that overlays rich data feeds on their views of the event. Each of these views is highly individualized, and may include information about parade floats, commentary from other spectators, video or text conversations with friends, and data gathered from sensors along the route. These devices give spectators access to parts of the parade only available through AR, and allow them to interact with parade elements in real time. Additionally, parade personnel, law enforcement, and paramedics utilize AR-enabled headsets to assist spectators through tasks, such as locating children who have been separated from parents or ensuring the safety and security of attendees by automatically monitoring any suspicious activities.

Providing rich experiences for users and assuring safety via support personnel requires processing a large amount of data while ensuring consistency and availability among end-users' devices and parade elements in the geographical vicinity of one another. If we conservatively estimate that the data feed from each AR-enabled device will require 10 Mbps of bandwidth, the large number of parade attendees could quickly saturate a 10 Gbps uplink to the Cloud. This data must be processed in a timely

manner, such that information associated with the object in view may be retrieved and overlaid on that object without perceptible delay or inconsistency to the user. If two users are looking at the same part of the parade route, it is important that they see the same common data displayed at the same time to enable a shared experience. Data associated with each object and user may also be privacy sensitive in nature, meaning that it may need to remain local to its source. For example, an attendee could use their personal device to locate a family member in the crowd via facial recognition, and local laws may require this tracking to be performed within the same region where it originates. When we consider that similar operations are being performed in tandem by thousands of users within the same area, we quickly realize that enabling such functionality without an agile Edge infrastructure can become untenable.

Enabling Cloud-like functionality across the Edge requires us to make inroads in several research areas. Developers have come to expect strong programming models and performance guarantees for the applications they create. To meet these expectations, we need to devise new methods of managing control and data planes to ensure timely resource placement decisions in a decentralized manner across heterogeneous micro data centers. At the same time, these orchestration efforts must maximize the efficiency of data storage and computation runtimes to ensure high utilization of limited Edge resources.

## PROGRAMMING MODELS

More than a decade of research into Cloud computing has provided us with robust programming models that allow for the seamless development of applications for the Cloud. The reliability of the Cloud allows these models to abstract away the need to account for issues such as backend failures or latency and bandwidth limitations between services supporting application components. This contrasts with development for the Edge, where creating effective applications currently requires several careful considerations. The placement and orchestration of application components at the Edge is difficult to perform manually, requiring domain expertise beyond the purview of most developers. At the same time, it is equally difficult to perform these operations in an automated fashion without some strong notion of application semantics. To address this problem, we must provide simple programming models that either implicitly or explicitly expose enough semantic information without creating an undue burden on developers.

An Edge-centric programming model must hide all the complexities of where and how the program components should be executed to achieve optimal performance. For example, the programming model provided by the Apache Spark cluster-computing framework [4] allows developers to harness the power of big data without needing to understand the intricacies of placement and availability in a distributed computing environment. We must support the same degree of functionality for developers at the Edge by providing programming idioms for a geo-distributed setting, which we describe below.

**Composition & Synthesis of Application Pipelines:** In our motivating scenario, users rely on video streams to enhance their experiences. One enhancement to this experience is the ability to identify and track parade objects. For example, a spectator may wish to be notified when the camera sees their favorite celebrity, and then retrieve information about the float on which the celebrity is riding.

This operation requires performing image recognition on video frames from the AR device to identify the celebrity's current position and then tracking the celebrity and float across the spectator's field of view. A programming model at the Edge could provide a declarative framework with a SQL-like syntax to allow developers to express their intentions for tracking objects via image recognition without the need to understand the specifics of how the task is performed. The declarative framework provides a way for users to intuitively compose queries for

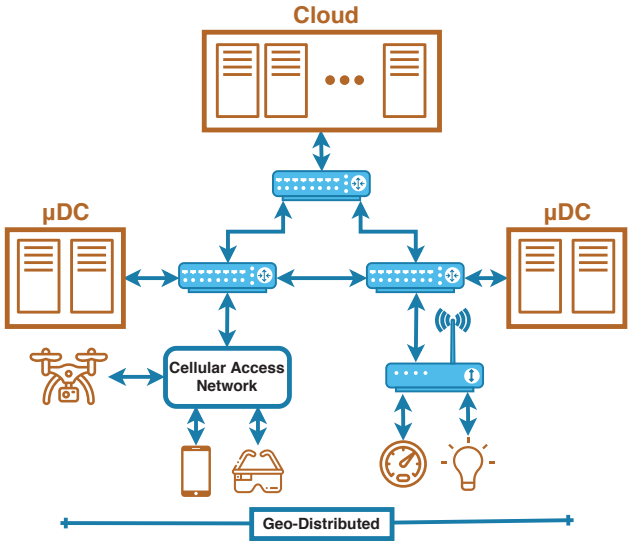


**FIGURE 1.** An Edge Computing ecosystem comprising mobile devices, IoT platforms, gateways, micro data centers, and the Cloud.

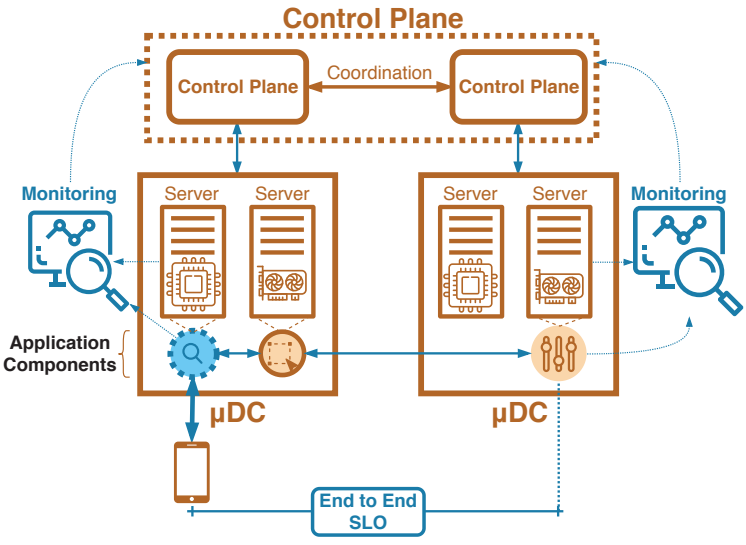


**FIGURE 2.** An example orchestration framework, wherein monitoring data is aggregated in individual micro data centers and shared among them to make coordinated deployment and migration decisions to meet application SLOs.

desired outcomes. The framework would automatically synthesize the query into a Data Flow Graph (DFG) where each node is a function that must be executed in a pipelined manner to service the query. In this way, domain experts in fields such as computer vision can guide the synthesis of complex application components without needing to be directly involved in the application development. Similarly, app developers can benefit from such domain expertise without the need to learn new techniques or algorithms beyond their sphere of knowledge. A generic declarative framework for all use cases would neither be feasible nor appropriate given the diversity in the application landscape. Instead, we envision a domain-specific approach that allows the synthesis stage to convert an SQL-like query into a DFG sharing generic application components from a shared library while giving the latitude to the domain expert and/or the app developer to specify user-defined components to be included in the execution pipeline.

**Performance Guarantees Through Service Level Objectives:** From a developer's perspective, the unaided deployment of applications on an Edge computing infrastructure is a daunting task. Applications have diverse performance requirements, and to ensure these requirements are met the developer would need to keep track of variables such as the locations of both clients and micro data centers, the network connectivity status of these micro data centers, and the current resource capacities and utilization of Edge sites. These requirements are exacerbated for next generation applications characterized by high device mobility and the consequent churn in resource usage at an Edge site. To alleviate this burden, developers should instead be provided with a way to express their application performance requirements as a series of Service Level Objectives (SLOs), which provide hints to the resource orchestrator about each application's requirements. The SLOs must specify end-to-end requirements so that all sources of performance overheads are considered during the orchestration of application components across a heterogeneous infrastructure. An example of such an SLO would be the *latency lower-bound* needed to provide real-time tracking of events in the parade and presenting such events to the user. If an application's SLO cannot be met (e.g., due to lack of resources at the Edge), we rely on a combination of access policies, priority of operations, and graceful degradation of services to provide as much of the requested functionality as possible. An access policy dictates whether new incoming requests are allowed, while priority of operations determines the order in which existing users will have their workloads processed. Where possible, a service may be gracefully degraded by measures such as decreasing camera frame rates or switching to a less accurate machine learning model to reduce resource utilization at the expense of achieving outputs of the highest quality. This method allows us to continue to provide some level of service until adequate resources become available again.

**Failure Handling:** When working within Edge computing environments, we cannot assume the availability of consistent power, cooling, or connectivity. The framework should provide a strong failure handling model in the presence of failures to ensure recovery and completion of the elements of the application components in the DFG. For example, if one or more Edge sites disappear during the pipelined execution of an image processing operation, the framework should recognize this failure and redistribute workloads to available Edge sites while updating graph dependencies.

## AUTONOMOUS ORCHESTRATION

Applications expressed in an Edge programming model need to be executed on the limited resources of a heterogeneous Edge infrastructure while maintaining SLOs. A control plane orchestrator responsible for monitoring the health and resource utilization of Edge sites is needed to make informed scheduling decisions. Several components are needed to make such an agile control plane a reality.

**SLO-aware Deployment and Adaptive Reconfiguration of Applications:** The orchestrator should take specifications synthesized by the programming model and use them to inform its decision on placing application components throughout the Edge. Examples of specifications include end-to-end performance requirements and dependency on specialized hardware devices. Application placement would also need to consider infrastructure-specific goals such as reliability, load balancing to avoid hotspots, and ensuring high utilization of scarce edge resources.

The high degree of dynamism (e.g., due to client mobility or temporary skews in workload) at the Edge necessitates the continuous adaptation of applications to meet SLO guarantees and maintain efficient usage of resources. Continuously monitoring the performance of deployed applications is required so that reconfiguration decisions are performed in an agile manner. The orchestrator maintains a reconfiguration policy to determine the appropriate actions for each application instance. Such operations can range from scaling application components both horizontally and vertically as well as enabling the live migration of applications commensurate with client mobility.

**Control Plane Decentralization:** Some Edge communications must pass through public infrastructure, which will decrease the perceived reliability of the network compared to Cloud data centers. To achieve high availability, the control-plane must either be decentralized or federated, such that multiple autonomous components in charge of different parts of the infrastructure can provide resources efficiently and independently, even in partial-disconnection scenarios. As an added benefit, decentralization also makes continuous monitoring of applications and resources more scalable.

Additionally, decentralization forces the use of efficient mechanisms for tracking the state of partitioned resources while simultaneously achieving globally optimal placements. Ideally, this state would appear as a single consistent view to orchestration algorithms in each micro data center and applications. One possible implementation of the orchestration framework is illustrated in Figure 2. In this diagram, distributed monitoring components track the health and performance of both micro data center hardware and application components and report these statistics to the distributed orchestrators. These orchestrators coordinate among each other to share monitoring information and use this knowledge to make intelligent management decisions guaranteeing end-to-end SLOs.

## APPLICATION STORAGE

The typical way of handling application data in the Cloud is by leveraging platform services like key-value stores and publish-subscribe systems, such that the manage ment of data is abstracted away from the developer. These services were designed and optimized for a Cloud data center environment, where strong connectivity and reliability are assumed to be readily available. Adapting such services for the Edge requires a change in design to account for situations where these assumptions may not hold.

**Data Partitioning for Low Latency:** Data partitioning in key-value stores like Cassandra [5] and publish-subscribe systems like Pulsar [6] fundamentally rely on selecting the best server to host a partition of data. Enabling such functionality at the Edge is possible by creating data store systems where a developer may provide constraints on data access latencies used to determine the optimal set of Edge sites for hosting each client's data, as illustrated in Figure 3. User mobility could result in violating the latency constraints specified by the developer. Upon noticing such violations, the system would transparently relocate the data to an appropriate Edge site based on proximity of the end-device.

**Consistency vs. Fault-Tolerance Tradeoff:** The higher failure probability of Edge infra-structure requires the use of replication to ensure fault-tolerance, and thereby introduces novel challenges for maintaining consistency among replicas. Providing stronger guarantees than eventual consistency is conducive to faster development time and fewer bugs. [7] Strongly consistent operations require multiple roundtrips between replicas, and therefore low-latency data access necessitates that all replicas be in close proximity to each other. Unfortunately, doing so leads to an increased vulnerability to correlated failures. [8] One way to cope with this tradeoff is through novel consistency models as introduced by Gupta, et al. [9] and Saurez et al. [10] The consistency semantics in these works allow the system to perform geo-replication with a reasonable degree of fault-tolerance and lower overhead than traditional geo-replicated databases, while providing low latency access for data of immediate interest as shown in Figure 3.

**Handling Skews in Workload:** Spatial and temporal skews in workloads can lead to hotspots which can adversely impact the tail latency of responses. [11] One way to address this problem is through data distribution schemes that are a hybrid of (1) schemes that partition data for low-latency, and (2) schemes that provide even load distribution. One approach for key-value stores is proposed by Gupta, et al. [12], wherein the partition key of a data item is calculated using its location and a consistent hash of its timestamp and item-type field. Another approach worth considering is the live migration of data away from overloaded Edge sites in order to reduce the load on them and handle skews of a transient nature.

The above approaches to addressing the challenges of storage at the Edge should be integrated to create an overall storage architecture. Such an integrated architecture would seamlessly provide the look and feel of a fast and reliable storage while abstracting away all the intricacies of the Edge environment from the developers.

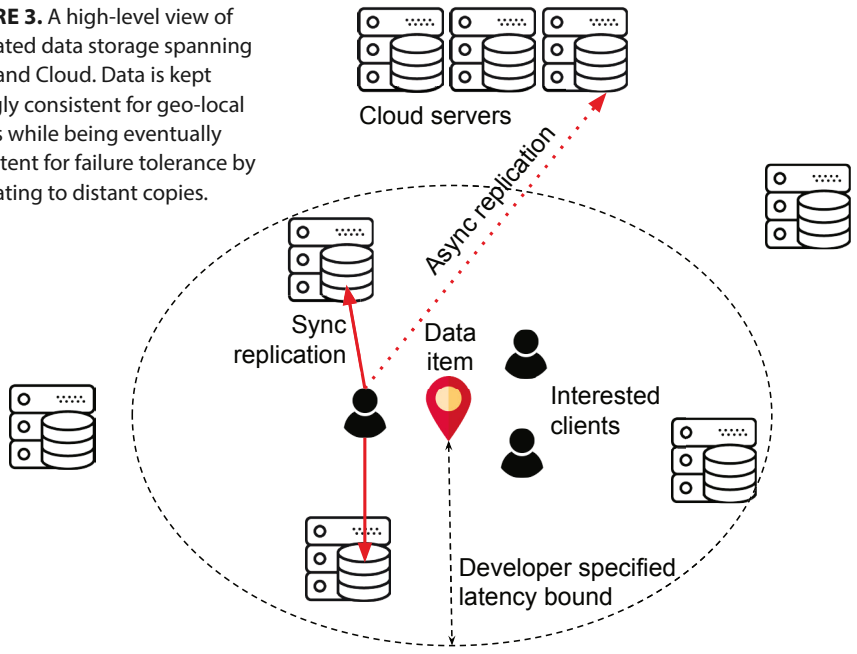


**FIGURE 3.** A high-level view of replicated data storage spanning Edge and Cloud. Data is kept strongly consistent for geo-local clients while being eventually consistent for failure tolerance by replicating to distant copies.

## APPLICATION EXECUTION

In a traditional Cloud model, we assume the notion of unlimited computing power. This assumption allows us to provision continuously running dedicated virtual machines or containers for hosting each application. In contrast, limited computing resources at the Edge make it impractical to perform such provisioning in every situation. To enable the high degree of multi-tenancy needed to support a variety of applications, we must devise new paradigms for the runtimes which execute applications.

If we consider our motivating scenario, we can see that different application components will have different computational lifetimes. For example, if the social media platform is performing a large amount of image recognition processing while emergency personnel look for a missing child, it would make sense to devote a longer running container to these operations during periods of high activity. In contrast, if content generation from users is infrequent during the same period, it would make sense to serve these operations on an ad-hoc basis. These patterns suggest we can achieve a higher degree of efficiency by adopting a hybrid model for execution environments that hosts applications commensurate with their needs.

One method for hosting applications on an ad-hoc basis comes from the notion of Function-as-a-Service (FaaS). In the FaaS model, applications exist as single purpose functions which only execute for a limited period of time when they are called and then shut down until needed again. Each function is hosted in a separate container,

which is instantiated upon function invocation and destroyed after a brief period of inactivity. Since application components only exist on an as-needed basis, we avoid unnecessarily committing resources and in turn allow a greater degree of sharing for those resources. The FaaS computing model allows us to achieve the much higher degree of multi-tenancy needed to serve a large number of clients.

Longer running applications would best be served by the more traditional method of hosting in containers. Containers provide a lightweight mechanism for segmenting operating system components and limiting resource consumption specific to one or more processes. However, we still must ensure these containers are properly adapted for Edge computing scenarios. For example, situations may arise where long-running applications must be temporarily pre-empted to serve spikes in demand for short-term applications. Ensuring that we may safely make these trade-offs allows us to provide many of the benefits of a traditional hosting model when necessary while still maximizing the utilization of limited resources.

In both scenarios, opportunities for innovation exist in two ways. First, we can improve container runtimes to meet the requirements of the Edge. The runtimes in state-of-the-art container frameworks include a broad array of functionality, which may not be needed in an Edge computing environment. Such unnecessary functionality adds overhead, which manifests itself in issues such as the *cold start problem*, where container instantiations introduce delays of 300 ms or more before a function can begin execution. These delays destroy the low-latency advantages of hosting applications at the Edge and must be reduced or eliminated. The second opportunity for innovation exists in developing new runtime methods for executing applications. For example, Hall et al. [13] discuss the use of the *WebAssembly binary format* as a way to provide strong isolation and resource provisioning mechanisms to functions without the overhead of containers.

A hybrid runtime could leverage innovations from both approaches to provide low-latency application executions. For short-running applications, a lightweight model such as WebAssembly could be used to provide fast loading and portability. For longer running applications, the traditional container-based model could be used. The execution platform could switch between serving applications with either model based on their recent characteristics.

## CONCLUSION

The task of elevating the Edge to be a peer of the Cloud is not without challenges. However, it is important to meet these challenges if we wish to enable the next wave of mobile applications. These applications all share common requirements for dynamic scalability, low-latency communication, and efficient in-network processing to provide the *Sense-Process-Actuate* workflow used in dealing with real-world data streams, suggesting that there is a common solution to meet their needs. The notion of Edge computing seeks to extend the utility of the Cloud to the last mile of the network. Its goal is to provide this utility through resources that are hierarchical and geo-distributed in nature. To enable an autonomous Edge, horizontal peer-to-peer interactions among Edge sites is essential. This need stems from the fact that different interacting client devices may be connected to different Edge sites at the same time. Such interactions are made possible by programming model, orchestration, storage, and execution paradigms which are specifically tailored to the unique challenges of the Edge. By building these paradigms, we elevate the Edge to be a peer to the Cloud, in turn creating the next evolution in mobile computing. ■

**Umakishore Ramachandran** received his PhD in Computer Science from UW-Madison in 1986 and has been with Georgia Tech ever since, where he is currently a Professor in the College of Computing. His research interests include parallel and distributed systems with an emphasis on Edge computing. rama@gatech.edu

**Harshit Gupta** is a PhD candidate in Computer Science at the Georgia Institute of Technology. He received his bachelor and master's of Technology in Computer Science and Engineering from IIT Kharagpur, India in 2016. His research is focused on orchestration and data management in Edge computing. hgupta@gatech.edu

**Adam Hall** is a PhD candidate in Computer Science at Georgia Tech. His research focuses on exploring new runtime methodologies for Serverless and Edge computing environments. ach@gatech.edu

**Enrique Saurez** is a PhD candidate in Computer Science at Georgia Tech. His research interests include distributed systems, data processing frameworks, and edge/geo-distributed computing. esaurez@gatech.edu

**Zhuangdi Xu** is a PhD candidate at the College of Computing, Georgia Tech. His research interests include camera networks and storage systems for Edge computing. xzdandy@gatech.edu

## REFERENCES

[1] Microsoft, "Azure IoT Edge," [online] https://azure.microsoft.com/en-us/services/iot-edge/.

[2] Amazon. "AWS for the Edge." https://aws.amazon.com/edge/

[3] Forbes, "The Macy's Thanksgiving Day Parade 2016 By The Numbers," Nov. 23, 2016. https://www.forbes.com/sites/hayleycuccinello/2016/11/23/the-macys-thanksgiving-day-parade-2016-by-the-numbers.

[4] Apache Software Foundation, "Spark Programming Guide." https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html.

[5] Apache Software Foundation. Apache Cassandra. https://cassandra.apache.org

[6] Apache Software Foundation. Pulsar. https://pulsar.apache.org

[7] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina and S. Ellner. 2013. F1: A distributed SQL database that scales, in *Proceedings of the VLDB Endowment.*

[8] R.S. Couto, S. Secci, M.E.M. Campista and L.H.M. Costa. 2014. Latency versus survivability in geo-distributed data center design, in *IEEE Global Communications Conference.*

[9] H. Gupta, & U. Ramachandran. 2018. FogStore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access. *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems.*

[10] E. Saurez, B. Balasubramanian, R. Schlichting, B. Tschaen, Z. Huang, S. P. Narayanan and U. Ramachandran. METRIC: A middleware for entry transactional database clustering at the edge, in *Proceedings of the 3rd Workshop On Middleware for Edge Clouds & Cloudlets, 2018.*

[11] S. Khare, H. Sun, K. Zhang, J. Gascon-Samson, A. Gokhale and X. Koutsoukos. 2015. Ensuring low-latency and scalable data dissemination for smart-city applications, in *Proceedings of the Tenth European Conference on Computer Systems.*

[12] H. Gupta, Z. Xu, & U. Ramachandran. 2018. DataFog: Towards a holistic data management platform for the IoT age at the network edge. *USENIX Workshop on Hot Topics in Edge Computing.*

[13] A. Hall & U. Ramachandran. 2019. An execution model for serverless functions at the edge. In *Proceedings of the International Conference on Internet of Things Design and Implementation.*