Simulating Edge Computing Environments to Optimize Application Experience

James R. Blakley[1], **Roger Iyengar**[1] **and Michel Roy**[2]

[1]***Carnegie Mellon University School of Computer Science***
[2]***InterDigital, Inc.***

November 2020
CMU-CS-20-135

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The emergence of edge computing introduces new complexity in the creation of distributed mobile applications. Application designers can now deploy application functionality in three or more tiers of compute infrastructure to optimize bandwidth, latency, cost, user experience and privacy for their users and their own operations. However, the diversity of edge and cloud computing resources, networks and end devices challenges the designer's ability to make efficient distribution choices. Tools to support this design task are in their infancy. This paper presents our work in creating a simulation framework for measuring and modeling edge computing infrastructure as a tool for application characterization. It uses a specially instrumented client application, the AdvantEDGE edge emulation platform, physical cloudlets and commercial LTE networks to gather application and infrastructure measurements that inform design decisions.

This research was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0051 and by the National Science Foundation (NSF) under grant number CNS-1518865 and the NSF Graduate Research Fellowship under grant numbers DGE1252522 and DGE1745016. Additional support was provided by Intel, Vodafone, Deutsche Telekom, Crown Castle, InterDigital, Seagate, Microsoft, VMware and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

Keywords: edge computing, edge-native applications, mobile networks, network emulation

1 Introduction

Edge computing [1] brings the promise of enabling new *edge-native* applications [2] that need low latency and high bandwidth connections to mobile and wired edge devices to achieve acceptable user experience. It has been shown [3] that without edge computing, end-to-end network latency can exceed 150ms.[1] Many connected user devices such as cell phones, cameras, vehicles, etc., referred to as *user equipment (UE)* by the telecommunications industry, produce data volumes that exceed the viable economic costs and acceptable transport times to transfer from the UE to a remote cloud. The traditional approach to managing these challenges has been to deploy application functionality on the UE that mitigates the need to transfer data to the cloud. For example, traditional mobile gaming typically implements the majority of game functionality on the UE with only time-insensitive tasks implemented in the cloud. Similarly, smart cameras may perform cropping, downsampling and encoding functions on incoming streams in order to reduce the transferred bitrate.

These techniques can meet the needs of many applications but, for others, the application experience quality can be inadequate. In gaming, for example, lighting effects may be of low quality due to the processing limitations of the UE graphics processing unit (GPU). A computer vision application may become less accurate when a highly compressed bitstream is sent to the cloud for processing.

Edge computing offers a solution to these problems by reducing the physical network distance and the number of network hops from the device to nearby application computing resources. An *edge-native* application is one that is designed to take advantage of attributes that this closer placement provides: low latency, bandwidth scalability, privacy-preservation and WAN-failure resiliency. Figure 1 shows the architecture of a typical edge-native mobile application. In this model, the application developer can specify what functionality is deployed on the UE, the nearby edge computing *cloudlet* [4] and in the remote data center.

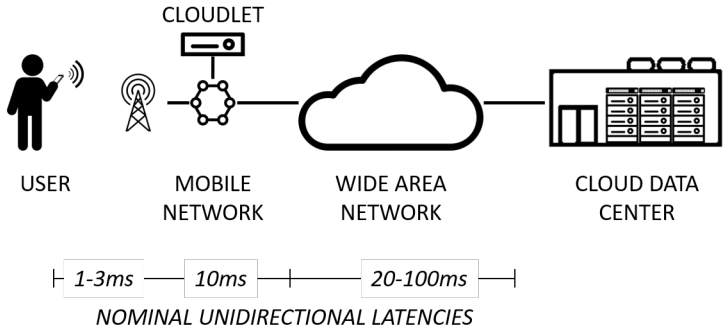

Figure 1: Mobile Distributed Application Architecture

The tradeoffs of this choice are shown qualitatively in Figure 2. Deployment design may be static (i.e., determined at design time) or dynamic (i.e., determined during runtime). In a dynamic model, the application responds to changes in operating characteristics by a) throttling certain aspects of the application to reduce its load on the end-to-end system or b) moving functionality from one tier to another. Content delivery networks (CDNs) [5] provide simple examples of both

[1] This number excludes the application processing time at the client and server.

cases. The "static" CDN design moves large scale video streaming functionality from a central video repository to video caches distributed to network edges. Then, at runtime, in response to poor throughput to UEs, these video caches use techniques like HTTP Live Streaming (HLS) [6] and MPEG-DASH [7] to dynamically reduce the transmitted bandwidth from the edge node to the UE (at a reduced video quality but with fewer streaming delays). More complex and performance sensitive applications have more difficult design choices. For example, an edge-native multiplayer game must optimize for individual user experience and for player interaction experience.

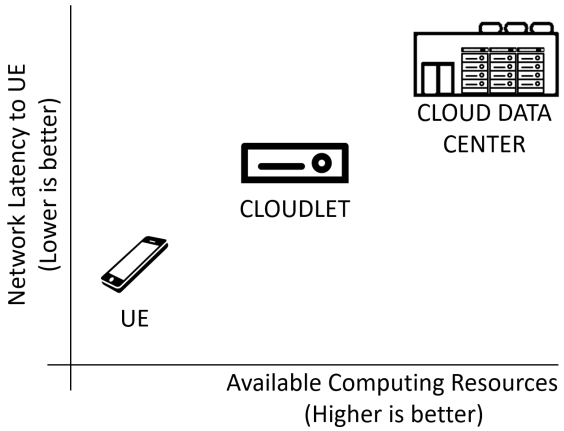

Figure 2: Application Design Tradeoffs

These choices are significantly complicated by several infrastructure related factors:

- Commercial mobile networks are diverse and their topologies and characteristics are generally opaque to application designers.
- Edge computing networks are new, non-standardized and sparsely deployed at this time.
- The resources available at each cloudlet are limited and shared with other users.
- Different mobile UEs have widely varying capabilities and features.

This report introduces a simulation framework that can provide edge-native application designers key insight for making static and dynamic deployment decisions. It can also provide edge computing network designers with insight into cloudlet sizing, placement and interconnect. This framework is then used to evaluate four edge-native application challenges:

1. Emulating different carrier interexchange architectures to simulate how cloudlets and interexchange point placement affects application experience.
2. Simulating the impact of cloudlet placement on application performance.
3. Characterizing existing commercial mobile networks to understand how "black box" real networks impact application experience.
4. Visualizing virtual scenarios using the real world characterizations.

The report is structured as follows:

- Section 2 gives more background and related work.
- Section 3 describes the simulation framework.
- Section 4 shows the simulation results of the above scenarios.
- Section 5 discusses these results, conclusions and areas for further work.

2 Background and Related Work

This work builds on research and industry efforts in several areas.

- **Edge-native applications** are the raison d'être for edge computing. Understanding how these applications operate in edge computing networks is fundamental to designing both the applications and the networks that support them.
- **Edge Network Optimizations** are static and dynamic methods for allocating work between UE, cloudlet and cloud. In particular, others have, like us, used simulation and emulation to assist in these decisions.
- **Edge and Mobile Network Measurement** gathers the key metrics from real and simulated environments to enable the monitoring, analysis and control of those networks.
- **Application Quality of Experience Measurement** is the method for determining whether a system provides an adequate experience to its users.

2.1 Edge-Native Applications

The literature and press call out many applications across many industries that can benefit from edge computing networks. [2] defines the concept of edge-native applications and points to a number of specific edge-native applications. When examined by their characteristics, most applications fall into one of the following four categories. See Figure 3.

1. **Single User Interactive** – These applications involve a single user interacting through a mobile UE with a distributed application service. Although many users may use the service simultaneously, interaction between users is negligible. Examples include many augmented reality applications like wearable cognitive assistance [8] and virtual desktop infrastructure. The user experience for these applications is generally measured by response time and visual quality. This report focuses on applications in this category.
2. **Multi User Interactive** – These applications retain many of the characteristics of single user interactive applications but add significant interaction between users. Examples include multi-player gaming and video conferencing. User experience is still measured by response time and visual quality but delivering acceptable performance is complicated by the potential for collaborating users to be serviced by different cloudlets and mobile carriers.

3. **Edge Analytics** – These applications involve data collection and processing from distributed UEs to gain understanding and insight that can drive operational action. Often, transferring raw collected data to a centralized location is cost or transfer latency prohibitive or is unacceptable due to privacy concerns. Examples include intelligent processing of surveillance videos and distributed federated machine learning [9]. User experience is driven by the cost and time to insight from gathered data.

4. **Internet of Things (IOT) Sensor** – These applications aggregate connections from many distributed sensor and actuator UEs to provide control or control-assist and data analysis and collection functions. Examples include autonomous vehicles and distributed traffic monitoring services. User experience is driven by the response time for control functions and the cost and time to insight for analytics functions.

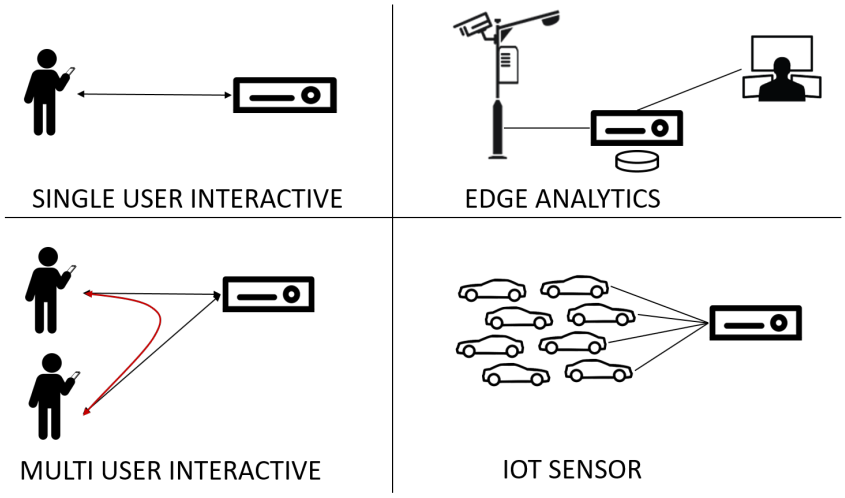

Figure 3: Edge-native Application Categories

This list excludes operator and operations related applications such as firewalls, traffic control and routing and other virtual network functions (VNF) [10]. It instead focuses on value-added services where an external consumer or business user gains a tangible and visible benefit from use of the service. While VNFs may exhibit similar characteristics to these applications (e.g., a virus scanning VNF may behave similarly to an edge analytics application), those applications were not considered as part of this work.

2.2 Edge Network Optimization

There is a large and growing body of research on allocating computing tasks across multi-tier device-edge-cloud application architectures. Much of this work focuses on scheduling-centric approaches that use dynamic network and application information to selectively execute atomic application tasks on different processing elements in the architecture [11], [12], [13]. Other work takes an empirical approach to optimize specific application types (cloud gaming, multimedia) at design time or run time (c.f., [14], [3], [15]).

This report focuses on the use of simulation and emulation to inform design or run time task distribution trade-offs. This approach can be used in concert with the other approaches. We take an application designer-centric perspective in the belief that, unless the designer can create an acceptable user experience, the overall efficiency of system wide optimization won't matter. A few others have used an approach similar to ours to combine simulation, emulation and application quality of experience optimization (c.f., [16], [17] and [18]).

2.3 Edge and Mobile Network Measurement

There is a long history of work in measuring the performance of networks in delivering quality of service to network users. Much of this work focuses on measuring and modeling the core network characteristics of latency, jitter, packet loss and throughput. This work provides a strong foundation for understanding how networks behave and can be readily applied to simulation design and execution (c.f., [19], [20], [21] and [22]). Many of the basic models from these works are applied in our simulations and in the AdvantEDGE emulation platform that we used.

2.4 Application Quality of Experience Measurement (QoE)

Application Quality of Experience [23] is the measurement of how well the user perceives the application to be performing. QoE measurement is application specific and can be quite difficult to define precisely. For most applications, a subjective QoE definition (e.g., visual inspection or Mean Opinion Score) must be transformed into an objective measure (e.g., Round Trip Time) to allow application design and test to be automated. This transformation requires the designer to determine which application characteristics drive the subjective measurement and define a specific set of metrics for those characteristics. QoE automation can also be complicated by an application's need for a human user to perform application tasks during the automation (aka, *a human-in-the-loop*). Some have created simulated users who randomly perform the required actions (c.f., [24]). This approach can greatly improve the scalability of simulations but make it difficult to correlate objective and subjective QoE measures.

3 Simulation Framework

Our simulation framework is depicted physically in Figure 4 and logically in Figure 5.[2] The framework provides an environment that models Figure 1 in a way that allows for experimentation. This section describes this framework while Section 4 shows the results of using the framework to test various scenarios. The framework consists of the following components:

- Section 3.1 Physical Infrastructure
- Section 3.2 Network Emulation Platform
- Section 3.3 Instrumented Client Application

[2]In this report, the term *simulation* means the execution of a test scenario on the framework described in this section. *Emulation* means the use of AdvantEDGE platform to emulate a mobile wireless network.

- Section 3.4 Simulation Automation Engine
- Section 3.5 Data Management and Analysis System

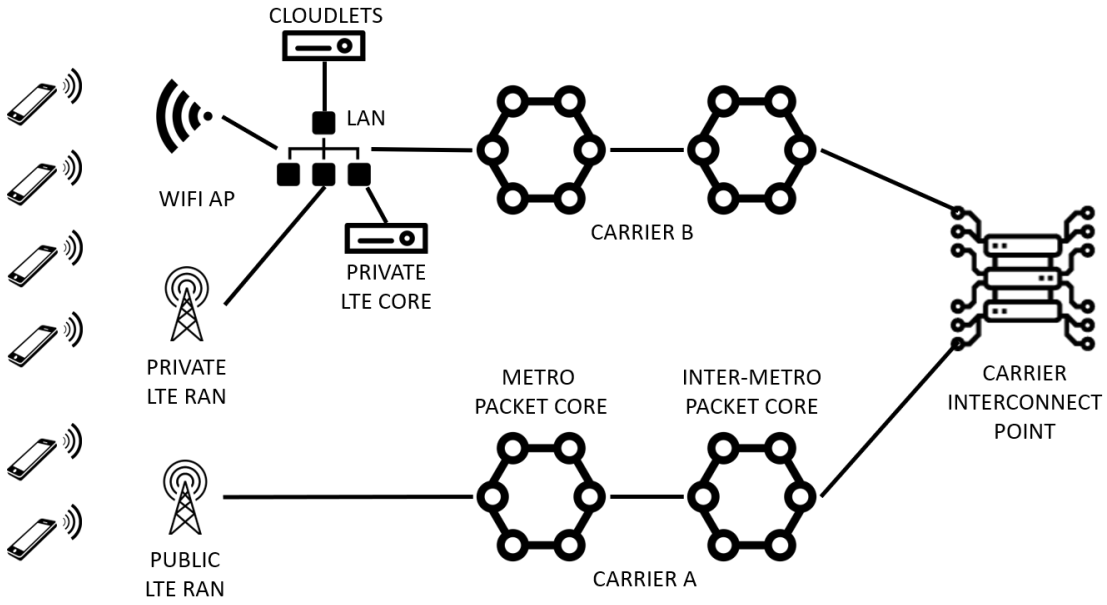

Figure 4: Framework Physical Architecture

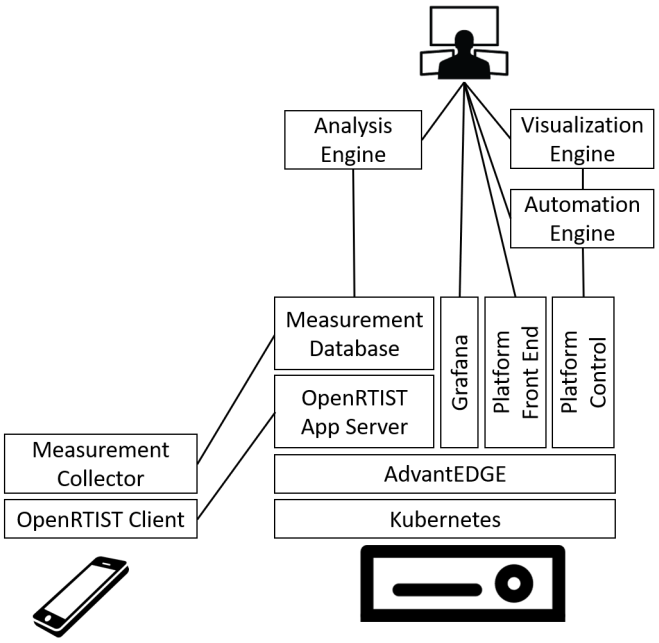

Figure 5: Framework Logical Architecture

The following sections provide a description of each component.

3.1 Physical Infrastructure

The physical infrastructure for the simulation framework is built on the Living Edge Lab [25] infrastructure at Carnegie Mellon University in Pittsburgh, Pennsylvania. It consists of three independent but interconnected wireless networks, a cloudlet and a set of mobile UEs. The three wireless networks are:

1. An in-builiding WiFi network connected to a wired LAN network. The cloudlet is also connected to the same wired LAN.

2. An outdoor private LTE network that is directly fiber connected to the wired LAN. The Private LTE wireless core resides on the same wired LAN.

3. Local commercial public LTE networks from AT&T and T-Mobile that are connected through a remote commercial interexchange point to the LAN.

Due to technical issues, the simulations described in Section 4 were done using the WiFi and public LTE networks.The WiFi network was used for the purely emulated testing as it introduced the minimum real network latency to the end-to-end application pipeline (<3ms). The public LTE network was used for the real world measurement cases. Using this network had the advantage of providing measurements from an operational commercial network, however the lack of local breakout meant that traffic between client and cloudlet travelled out of the metro area and, therefore, experienced an additional 20-40ms of one way network latency.

The cloudlet is a single node Intel® Core™ i7-6700 CPU @ 3.40GHz with an NVIDIA GeForce GTX 1060 3GB GPU.The server-side simulation framework and test application run on the cloudlet. The client UEs are android smartphones including a Samsung Galaxy S8 and an Essential PH-1.

3.2 Network Emulation Platform (AdvantEDGE)

The simulation framework is centered around the AdvantEDGE platform [26]. AdvantEDGE is a mobile edge emulation platform that runs on Docker and Kubernetes. AdvantEDGE provides an emulation environment that enables experimentation with edge computing technologies, applications, and services. The platform facilitates exploring edge deployment models and their impact on applications and services in short and agile iterations.

AdvantEDGE enables the user to define scenarios that include:

- A network topology of cloudlets, clients, wireless points of access, zones and UE
- Network characteristics for each element including latency, jitter, packet loss and throughput
- Network and mobility events to change network characteristics and the location of UE and cloudlets during simulation run time

It allows the connection of real cloudlet and UE applications so that simulation can capture the impact of network design on application performance. It also supports event scripting, collection of measurements in an offline InfluxDB time series database and real time Grafana dashboards. This combination makes it a powerful platform for edge network simulation. These capabilities were all used in the scenarios discussed below.

Figure 6: OpenRTIST Style Transfer

3.3 Instrumented Client Application

In edge-native applications, the client application running on the UE directly provides the user experience. It has visibility to application experience degradation caused by end-to-end latency, jitter, packet loss and bandwidth constraints. It also has access to network and location information that can be highly valuable for network analysis.[3] In addition, the nature of the application itself defines the user experience metrics of importance. For example, latency only matters if the delay is perceptible by a user and materially affects the experience. A test application for assessing edge computing networks must have a real user experience and be able to measure important quantitative experience metrics.

In this work, we used a modified version of the OpenRTIST [27] application as our test application. OpenRTIST is a simple single user interactive application that captures user video, transfers that video to the cloudlet, performs image style transfer [28] on the video and sends the styled video back to the UE for display (see Figure 6). Network degradation impacts the experience by making the styled video appear jerky and lagging behind real time. This experience can be quantitatively measured using two metrics that are captured by the client, *round trip time (RTT)* and *framerate (FPS)*. These two metrics are inversely related to each another and are heavily impacted by network latency and packet loss. RTT is measured by time stamping each video frame leaving the client and detecting when the corresponding styled frame is returned to the client. RTT includes the round trip network latencies, the style processing time at the cloudlet and the transmit and receive times at the client.[4] To prevent frame buffers from overflowing due to these delays, frames are dropped by the application at various points. This frame dropping reduces the application FPS.

In addition to these two user experience metrics, other data was captured for use in monitoring and analysis of the system. They are summarized in Table 1.

Except for *traceroute*, this data is captured every 10 frames as the application runs. Round trip times and framerates are averaged over the interval. Traceroute is captured only at session start however was verified to be relatively stable over the course of a session. Aside from the user

[3] Access to location and network information requires specific android permissions and, therefore, this data is not collected in the standard OpenRTIST client

[4] Times for camera frame capture and frame display on screen are not included in the RTT.

Measurement	Type	Description	Typical Value
Round Trip Time	User Experience	Transit time for image frame from client to cloudlet and return	30-100ms
Frame Rate	User Experience	Frames transmitted per second	10 FPS
Traceroute	Network	The network hops between client and cloudlet	IP Address List; End-to-end time
Signal	Network	Signal strength for WIFI or LTE connections to the client. *Also includes CellID of the connected cell site*	-80dbm
Carrier	Network	LTE Network Provider	AT&T
Connection Type	Network	WiFi or LTE	WIFI
Server URL	Network	Cloudlet IP Address and Port	192.168.1.207:31001
Location	UE	Client GPS Coordinates	Latitude, longitude
Country	UE	Current country code	us
Manufacturer	UE	Client manufacturer	samsung
Model	UE	Client model	SM-G950U
Phone Type	UE	Client mobile technology	gsm
Session ID	Timestamp	Session timestamp set when client connects to cloudlet	2020-08-24-13-56-35

Table 1: Instrumented Client Measurements

experience metrics, signal strength and CellID, the remaining measurement fields remain constant during a session.

At each measurement point, the data is pushed into the data management system on the cloudlet for monitoring and analysis. The data management system timestamps each measurement on insertion into InfluxDB. This assures that the client and the emulation system measurements are synchronized.

3.4 Automation Engine

To simulate a realistic mobile network, the configuration and characteristics of that network need to vary over the course of a simulation. This need requires an automation engine to script these variations for the specific simulation. The AdvantEDGE platform allows the creation of *mobility events* and *network characteristic events* as simulation building blocks. It presents these capabilities through REST APIs following the OAS 2.0 specification (a.k.a. Swagger) [29]. These

APIs allow for various language specific binding through swagger-codegen [30]. For this project, we used the python bindings to create an automation engine.[5]

Mobility events allow the movement of UEs from one Point of Access to another, cloudlets from one zone to another and cloudlet services from one cloudlet to another. *Network characteristic events* allow the characteristics of individual nodes to be changed. The primary controllable network characteristics are *latency, latency variation a.k.a. jitter, throughput* and *packet loss*.

Simulations are scripted using a simple python structure as shown in Listing 1. This script sets the network initial conditions and then steps through test events to move *ue1* around the network and change *zone-o1* latency to 2 ms. The *waitafter* parameter specifies the delay in seconds before the next event.

Listing 1: Scripting Example

```
    'basic':{
            'name': 'testscenario',
            'application':'openrtist',
            'initial_conditions': {
                'UE': {'latency':0,'latencyVariation':0,
                            'throughput':1000,'packetLoss':0 },
                'POA': {'latency':3,'latencyVariation':1,
                            'throughput':1000,'packetLoss':0 },
                'ZONE': {'latency':5,'latencyVariation':1,
                            'throughput':1000,'packetLoss':0 },
                'OPERATOR': {'latency':5,'latencyVariation':1,
                            'throughput':1000,'packetLoss':0 },
                'UE-APP': {'latency':0,'latencyVariation':1,
                            'throughput':1000,'packetLoss':0 },
                'EDGE-APP': {'latency':0,'latencyVariation':1,
                            'throughput':1000,'packetLoss':0 },
                'SCENARIO': {'latency':1000,'latencyVariation':10,
                            'throughput':1000,'packetLoss':0 },
            },
            'exceptions':[
                {'name':'zone1-o1','latency':0,'latencyVariation':10,
                            'throughput':500,'packetLoss':1},
                {'name':'zone1-o3','latency':25,'latencyVariation':10,
                            'throughput':500,'packetLoss':1}
            ],
            'test_events': [
                    {'type':'MOBILITY','mover':'ue1','dest':'poa1-o1','waitafter':30},
                    {'type':'NETWORK-CHARACTERISTICS-UPDATE',
                            'name':'zone1-o1','latency':2,'waitafter':10},
                    {'type':'MOBILITY','mover':'ue1','dest':'poa1-o2','waitafter':40},
                    {'type':'MOBILITY','mover':'ue1','dest':'poa1-o1','waitafter':40}
                ],

        }
```

[5]AdvantEDGE has a native automation engine however this work began using AdvantEDGE V1.4 which did not allow scripting of network characteristics. That functionality was added to the native API in V1.5 but, rather than convert to that API, we upgraded our scripting capability to work with V1.5.

3.5 Data Management and Analysis

The value of a simulation is derived primarily from insights from the data collected. As mentioned above, the AdvantEDGE platform and instrumented client load data into an InfluxDB time series database. AdvantEDGE stores network and event measurements from deployed scenarios. The instrumented client stores the measurements described in 3.3. The database is accessible for:

- Display on AdvantEDGE and Grafana dashboards
- Extraction, analysis and visualization by any number of external analytics engines

We primarily use Grafana to monitor scenario execution and for demonstration purposes. We primarily used pandas, geopandas, numpy and matplotlib for analysis and visualization.

4 Simulation Scenarios

The simulation framework described in Section 3 was created to run simulations that would provide insight on key edge computing issues. In this section, we describe four simulation scenarios used to understand the following questions:

1. Edge applications may require a user on one mobile carrier's network to interact with a cloudlet service or user on another carrier's network. How does the location of the interexchange point between these two carriers impact the user experience? How close to the user should the interexchange point be?
2. Cloudlet placement to maximize user experience at a minimum carrier cost is complex multi-dimensional problem with limited known art to apply. Can insights and learnings be obtained by simulating various cloudlet placements in a realistic emulated network?
3. For most application developers, the mobile network configuration and characteristics are opaque. Is it possible to reverse engineer a mobile network using an instrumented client and edge service?
4. Can we visualize a realistic network simulation that reflects the known characteristics of an existing mobile network?

The following sections describe these simulation scenarios in greater detail.

4.1 Baselining the Environment

Prior to running the specific model simulations, we baselined three attributes of our environment:

1. Application latency (L_A) with an unconstrained network (zero network latency, jitter and packet loss and infinite throughput).
2. The quantitative impact that network latency increases have on application performance (round trip time and framerate).

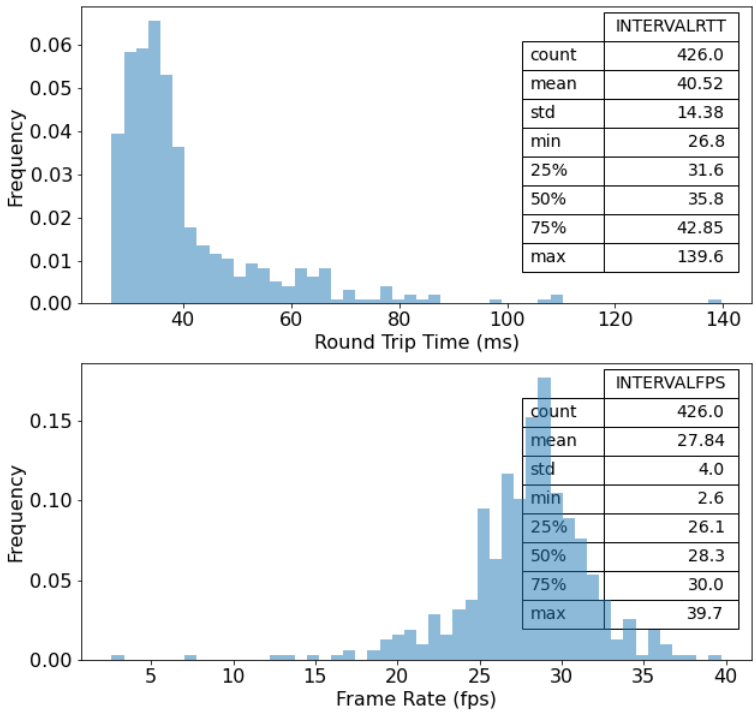

Figure 7: Application Only Round Trip Time and Framerate

3. The actual application performance on a commercial mobile LTE network.

A simple additive model would predict a round trip time of

$$RTT = \sum_i L_{u_i} + \sum_i L_{d_i} + L_U + L_C$$

where L_{u_i} is the latency of each upstream link, L_{d_i} is the latency of each downstream link, L_U is the latency of UE application and L_C is the latency of the cloudlet application (Total application latency, $L_A = L_U + L_C$). Running the simulation with the emulation network $L_u = L_d = 0$ gives a measured application latency, $L_A = 41 \pm 14ms$ as shown in Figure 7.[6]

To understand the impact of incremental network latency, we ran a simulation with monotonically increasing network latency in steps of 5ms for one minute with a range from 0ms to 50ms. The results of our first simulation are shown in Figure 8 which plots round trip time against network latency. The dots show the measured round trip time. The blue and green lines show linear and quadratic fits to the data while the orange line is the predicted latency from our equation above.

From this chart, we note:

[6] L_A also includes the latency added by the physical WiFi and ethernet network shown in Figure 4. The UE and cloudlet are directly connected to the same router and on the same subnet so we estimate this additional latency to be <5ms.

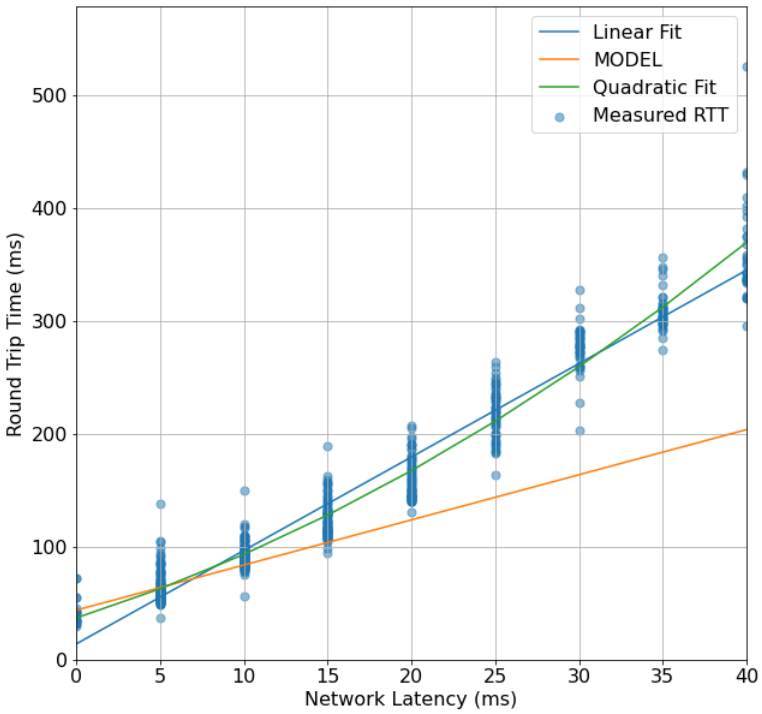

Figure 8: Anomalous Application Performance in Increasing Network Latency

- If network latencies were linearly additive to RTT, then we'd expect a 5ms network latency change to add 20ms to RTT ($2 \times \Delta L_{u_i} + 2 \times \Delta L_{d_i}$) to account for upload and download to both UE and cloudlet. For example, with a 41ms application only RTT, a network latency of 20ms should give a total RTT of 121ms.

- However, the data from this simulation shows a linear slope of 8.3 rather than the expected slope of 4.

- We also noted a slight unexpected non-linearity in the curve. The linear fit curve has an $R^2 = 0.9443$ while the quadratic fit has $R^2 = 0.9570$.

These anomalies led us to suspect an issue with the application or the simulation framework. We traced the issue to the use of Nagle's algorithm [31] on the the client-side TCP connection. Turning off Nagle's algorithm gave the expected results as shown in Figure 9.

The process described above highlights the utility of this simulation approach. Prior testing of the application outside the simulation framework did not reveal this impact of increasing network latency. Once it was identified, the subsequent solution showed a marked improvement in application performance.

We observed the effect of increasing latency visually on the application itself – as the network latency increases, the displayed video rapidly becomes choppy and delayed. Based on observation

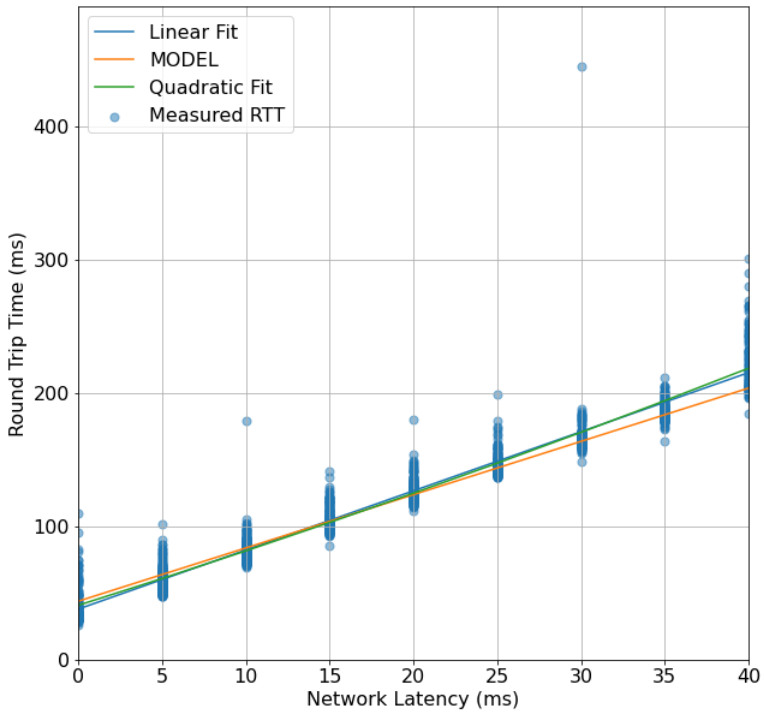

Figure 9: Expected Application Performance in Increasing Network Latency

of displayed video, the user experience becomes visibly degraded with an **RTT more than 150ms and framerate less than 10 FPS**. These thresholds are clearly subjective and application specific but, for the purposes of this report, were used as cut off thresholds for acceptable application performance.

We also baselined an expected RTT for a typical commercial LTE wireless network to assure that our network characteristics were in line with real world measurements. The data provided by our carrier partners provided a starting point for our model characteristics; our next step connected a mobile UE to the local Pittsburgh T-Mobile LTE network. The connection traversed T-Mobile to a remote carrier inter-exchange point (IXP) where it returned to Pittsburgh through the Verizon FIOS wired access network. The route of travel for a specific session is shown in Figure 10. We collected application performance data for this connection as shown in Figure 11. The mean RTT is 237ms and the mean FPS is 8 FPS – unacceptable application performance when compared with our acceptability criteria of 150 ms and 10 FPS.

4.2 Carrier Interexchange Simulation

In our work with the Open Edge Computing Initiative (OEC) [32], we were asked to help carriers understand the trade offs of different network positioning of carrier inter-exchange points (IXP) in the context of edge computing. This problem was first defined by Gerszberg [33]. IXPs are

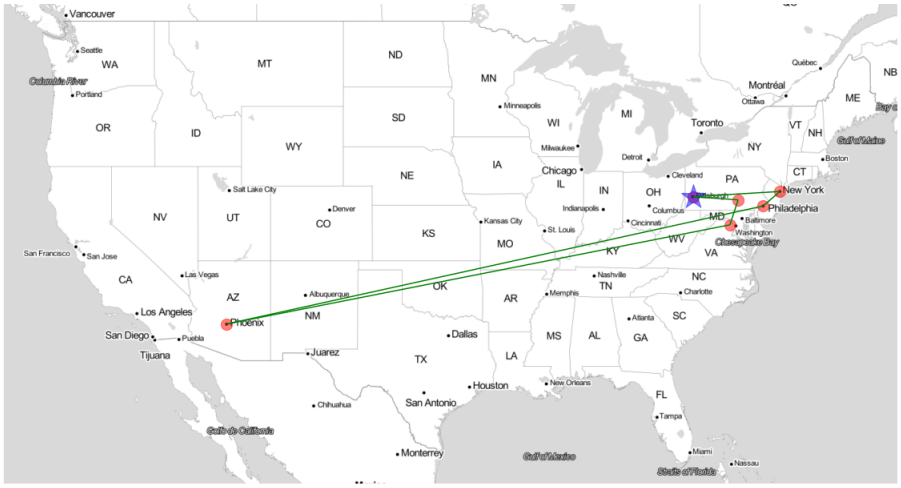

Figure 10: Mobile Wireless Traffic Route

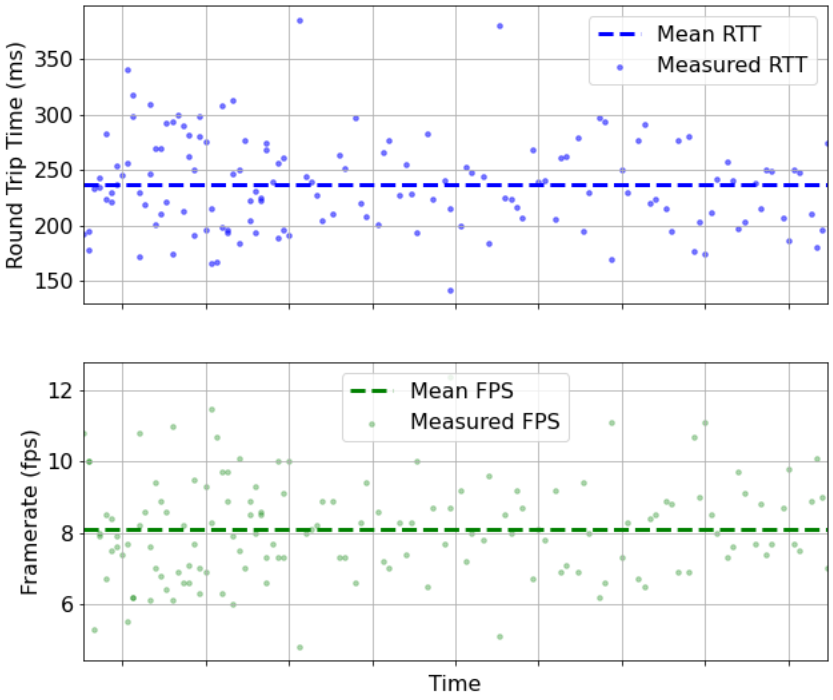

Figure 11: Mobile Network Application Measurements

physical locations at which carriers transfer user and network data that must move between carriers. In traditional networks, IXPs are often at centralized locations far from users and cloudlets. Data passing from a user on one carrier network to a cloudlet on another will need to travel across the first network to an IXP connecting to the second network before it can reach the cloudlet. Return packets will traverse a similar path in reverse. For multi-user interactive applications, traffic between users on different carrier networks will also need to pass through an IXP. Depending on the locations of the users, cloudlet and IXP, the added end-to-end latency can be 10s to 100s of

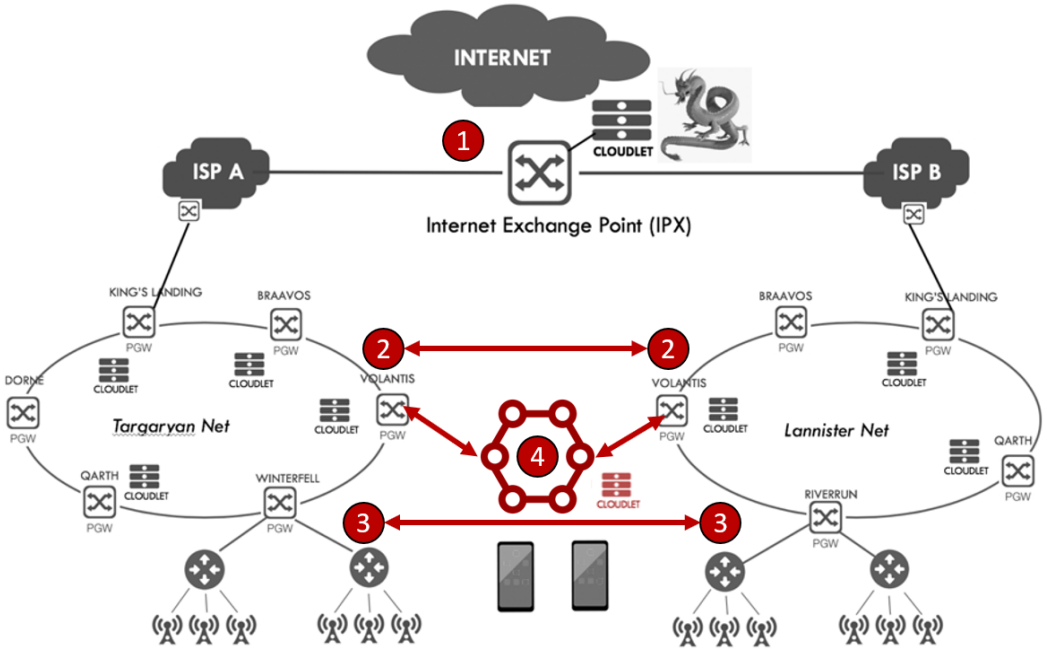

Figure 12: Potential Interexchange Points

milliseconds. Gerszberg defined four potential IXP positions for assessment as shown in Figure 12.

1. An IXP position geographically remote from the user and the cloudlet. This position is currently typical of many existing mobile networks and what we saw in network baselining in Figure 10. In the simulation, we assumed that the cloudlet was connected to the network at the radio access network (RAN), requiring packets from the cloudlet to traverse RAN, the metro core and the Wide Area Network (WAN) to reach the IXP.

2. An IXP position within the same metropolitan area (aka serving area) as the user and cloudlet. This position would typically be somewhere in the metro core infrastructure. In the simulation, we looked at two IXP locations in the metro core, a) at a point in the metro core far from the UE and cloudlet and b) at the edge of the RAN near the UE and the cloudlet.

3. An IXP position in the radio access network (RAN). This position puts the IXP very close to the UE and the cloudlet.

4. Two IXPs within the metro area core. In this case, the IXPs are provided by a third party neutral host who transfers the data between the two carriers. Based on partner input, we estimated that traversing two IXPs within the neutral host's metro network had $<$1ms impact on the latency as compared to Model 2.

Our scenario simulation goals were to measure the application user experience with users and cloudlets on different carrier networks given each of these IXP positions. We assume that costs increase as the IXP is moved closer to the network edge. This cost increase derives from the

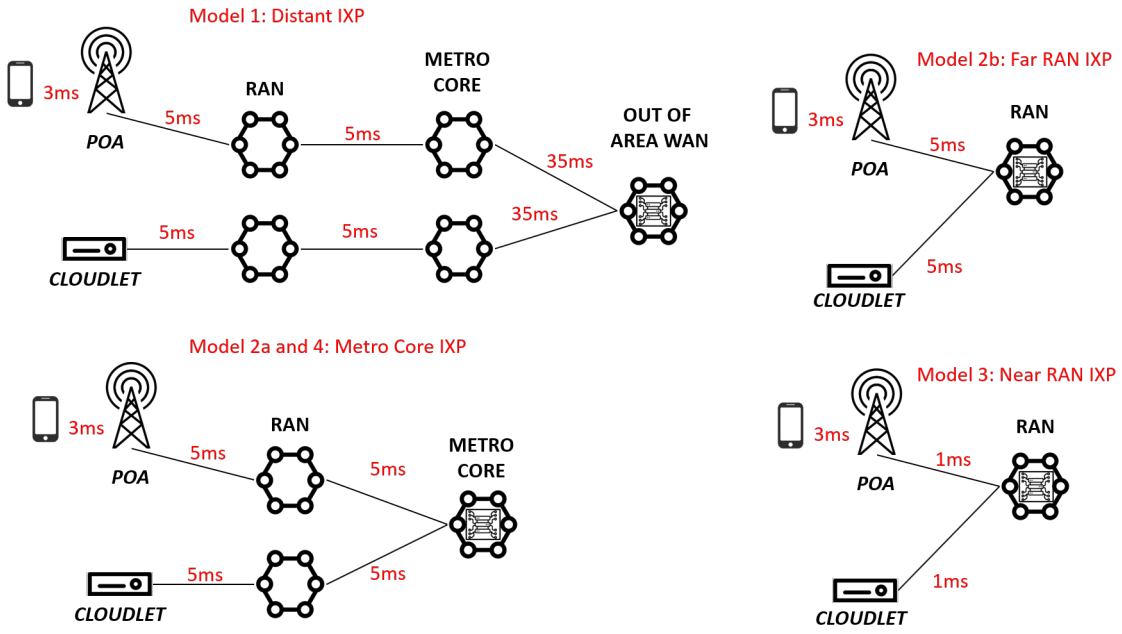

Figure 13: Simulation IXP Topologies

increased number of IXPs and increased transport infrastructure required to connect to the IXPs. Therefore, the optimal IXP position is the location furthest from the edge where the application user experience meets the minimum acceptable requirement. This criteria is obviously application specific and is complicated in multi-user interactive applications where the relative positions of users and cloudlets can be very complex.

To implement this simulation, we used the OpenRTIST instrumented client and created an AdvantEDGE network scenario that reflected the network topology and characteristics provided by OEC carrier members, especially Vodafone and VaporIO. Using OpenRTIST means that the results primarily reflect the single user interactive application case. The topology is shown in Figure 13 and 4G LTE network and application characteristics are shown in Table 2.[7]

Measurement	UE App	Wireless Link	RAN	Metro Core	Out of Area WAN	IXP	Cloudlet App
Mean Latency	Actual	3ms	5ms	5ms	35ms	1ms	Actual
Jitter	Actual	1ms	1ms	1ms	1ms	1ms	Actual
Throughput	Actual	1Gbps	1Gbps	1Gbps	1Gbps	1Gbps	Actual
Packet Loss	Actual	0	0	0	0	0	Actual

Table 2: Carrier Interexchange Simulation Characteristics

[7]For this simulation, thoughput and packet loss were set to values that simulated ideal conditions with no congestion or loss.

4.2.1 Distant IXP *(Base Case)*

With the baselining in Section 4.1 complete, we calibrated the Distant IXP simulation to align with the commercial LTE environment. As shown in Figure 11, the commercial LTE RTT averages 237 ms. We were able to calibrate our simulation to this value by setting the *out-of-area WAN* latency to 35ms (down from the 50ms estimate from our carrier advisors) and left the other characteristics unchanged.

The RTT and FPS measurements for all four models are shown in Figure 14. As we would expect from the baseline in Section 4.1, our Distant IXP simulation gives application RTT and FPS performance below acceptable per our criteria above. From this data and for this application, we can see that, the other models achieve acceptable application performance.

4.2.2 Metro Core and 3rd Party Metro IXP

The metro core model (Model 2) has two sub-cases. Case 2a places the IXP in the metro core near the connection to the out-of-area WAN. This IXP location would be typical as metro interconnect often occurs at the same physical location as carrier interconnect to the wide area internet. The metro core case 2b places the IXP in the metro core near the connection between the RAN and the metro core. The difference in the two cases is an incremental 5ms network latency between case 2b and case 2a. The 3rd party metro model (Model 4) is simulated with the same configuration as case 2a.

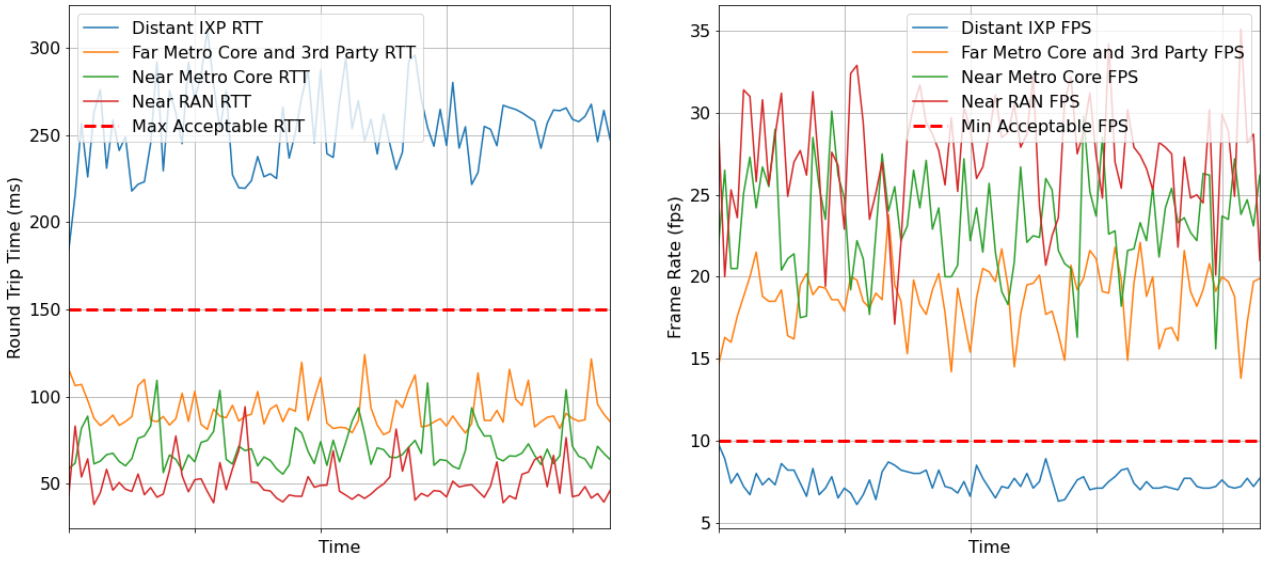

Figure 14: IXP Simulation Application Results

4.2.3 Near RAN IXP

The near RAN (Model 3) assumes that the IXP is placed very near the edge of the RAN such that the latency between the radio and cloudlet is less than 2ms. This model is the most expensive case as IXPs and cloudlets would necessarily be widely distributed and the network infrastructure

to create many distributed IXPs may be cost prohibitive. As seen in Figure 14, the incremental value to this application over the metro core and 3rd party metro models is slight. There may be applications (e.g., real time sensitive, safety critical IoT applications) where the added cost can be justified.

4.3 Cloudlet Placement Simulation

As we discussed the carrier IXP challenges with our OEC partners, they raised a more general question about placement of cloudlets in a single carrier's network. Cloudlet placement nearer to the user will, in theory, provide better user experience at a higher cost. To explore this question, we adapted the approach in Section 4.2 to examine three possible cloudlet placement points shown in Figure 15. The results of this analysis are shown in Figure 16. As can be seen, all placements

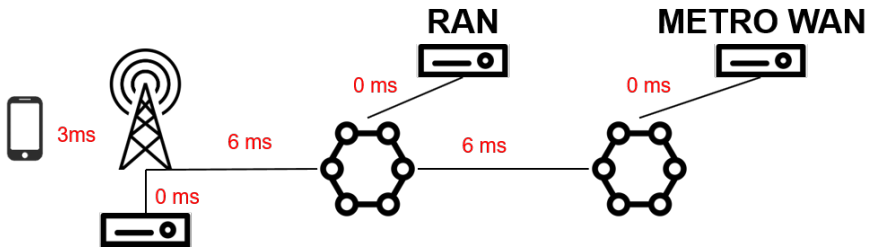

Figure 15: Cloudlet Placement in Mobile Networks

within the metro area give acceptable application performance.

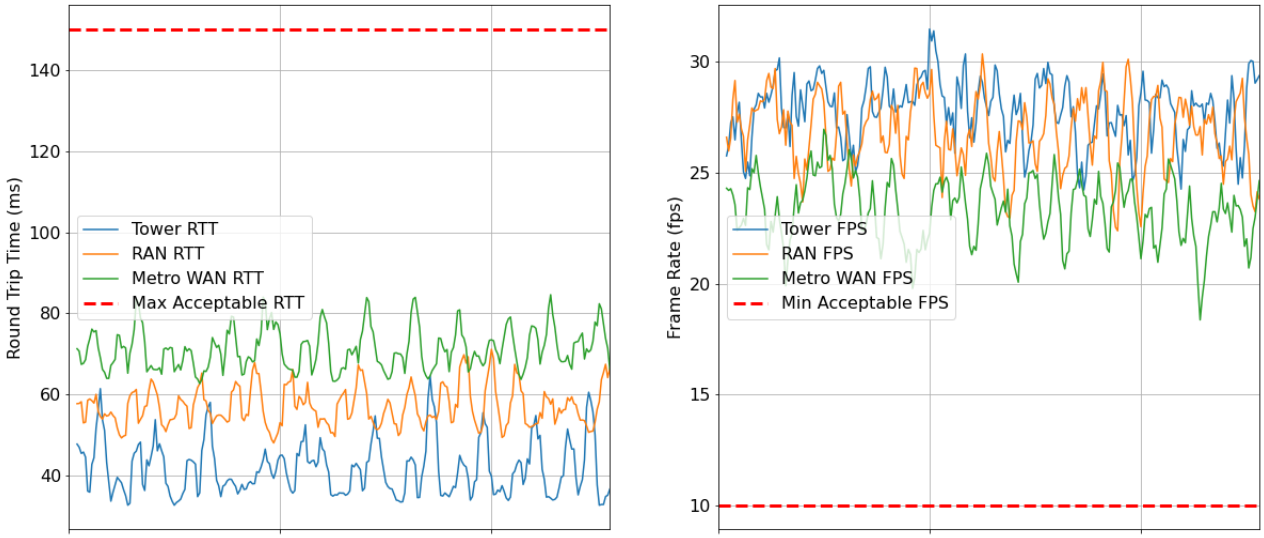

Figure 16: Application Performance at Different Cloudlet Placements

4.4 Mobile Network Characterization

One limitation of the IXP simulation in Section 4.2 is that it relies on network characteristics estimated from the experience of our carrier partners and the end-to-end baselining measurements in Section 4.1. It lacks detailed network topology and characteristics of the specific carrier networks that are simulated. These details are generally accessible only to internal carrier teams and are rarely shared across carriers. Accordingly, we used our best available assumptions and estimates in Sections 4.2 and 4.3. In this section, we ran a simulation that begins to give some insight in the nature of the local wireless network by analyzing data collected using the simulation framework.

As in our application baselining exercise in Section 4.1, we configured the AdvantEDGE platform to have no impact on the application by setting the emulated network to zero network latency, jitter and packet loss and infinite throughput. We connected the UE to the local T-Mobile or AT&T commercial LTE network then through a distant IXP to terminate locally on the Verizon FIOS network. We could then use the measurements collected by the instrumented client and the OpenCellid database [34] to characterize and visualize the local commercial network. Because this configuration still transits through a distant IXP, the measured RTT and FPS are dominated by the latency of that link. However, some insights can still be gained by studying the variation over time and UE location.

Data was collected by driving through Pittsburgh's East End around the Carnegie Mellon campus. The data collected is intended to be instructive rather than definitive and is limited to a single route on a single day with a specific client UE on a single commercial network. No conclusive information can be derived about the network from this limited scenario however it does show the possibilities inherent in the approach.

Figure 17 shows the route driven (blue dots) and the cell towers (orange triangles) that the UE connected to during the route. The size of the cell tower icon indicates the peak received signal strength from that tower. Orange lines show the connection to the cell tower for each measurement.

Some of the specific questions we sought to answer:

Question 1: *Does UE received signal strength or distance from cell tower affect application performance?*, Figure 18 shows the application performance for variable signal strengths. A linear regression produces an RTT R^2 of 0.0112. We conclude, for this limited dataset, that there appears to be no obvious connection between signal strength and application performance. We were somewhat surprised by this finding since signal strength does directly affect channel bandwidth. We can only conclude that the network was uncongested at the time of the test and a larger sample across time and location might lead to a different result. Figure 19 shows the application performance versus the distance of the UE from the cell tower. With an RTT R^2 of 0.0554, there also appears to be no obvious connection between cell tower distance and application performance. This is not a surprising result given the result of the signal strength experiment.

Question 2: *Is there a difference in application performance when the UE is moving versus stationary?* During the route there were a number of locations where the vehicle was stationary. A stationary location is defined as a location where more than 10 consecutive measurements were taken. In our sample of 2028 locations, 23 were stationary and 2005 were moving. Using a similar regression technique as above, there is no relationship between movement and application performance (RTT $R^2 = 0.0044$, FPS $R^2 = 0.0044$).

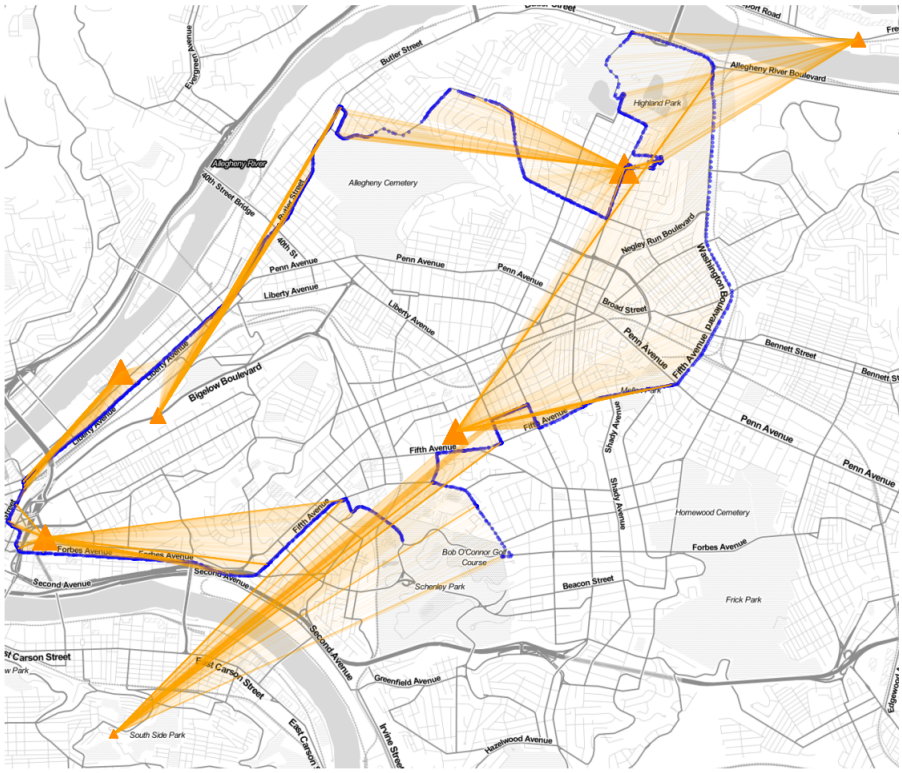

Figure 17: Route and Cell Towers

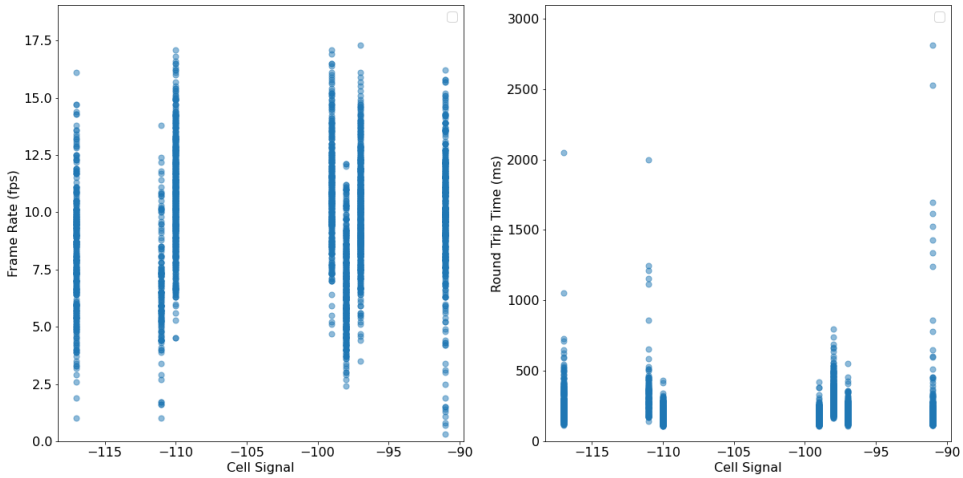

Figure 18: Application Metrics versus Cell Signal

Question 3: *Do handoffs between cell towers impact application performance?* During the route, there were a total of 6 handoffs between towers and 2843 measurements taken while connected to the same tower as the previous measurement. Figure 20 shows the application metrics for measurements with and without handoffs. The mean RTT was 13% higher during handoff than while remaining on a single tower. Results were also calculated using a handoff window including two measurements before and after the actual handoff. In this case, the mean handoff RTT was

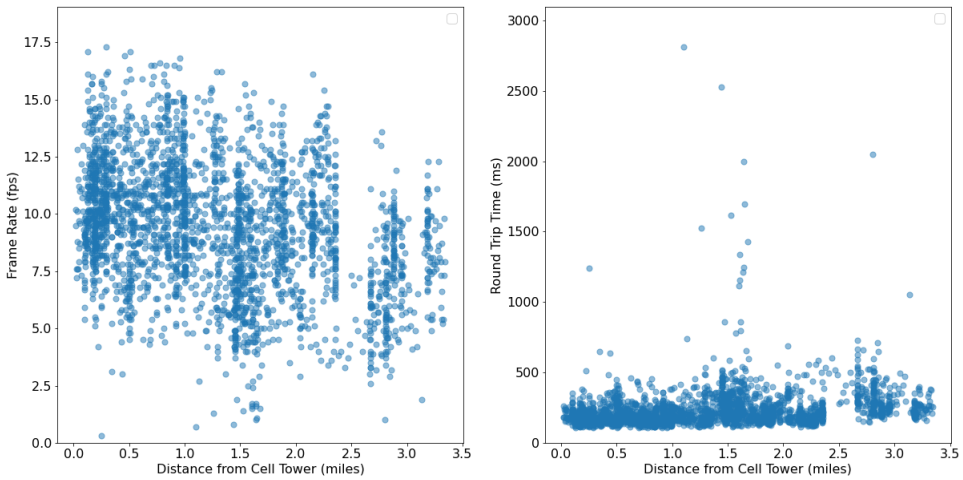

Figure 19: Application Metrics versus Distance

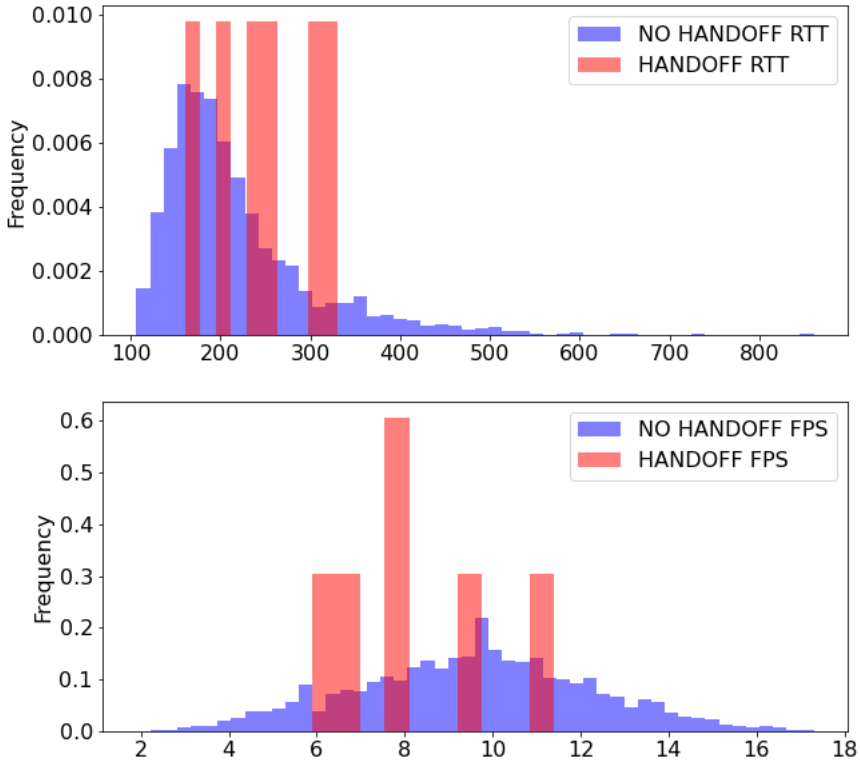

Figure 20: Application Metrics at Handoffs

19% higher than the no handoff case. While this result comes from a small sample size, it would appear that handoffs do cause a temporary loss in application performance.

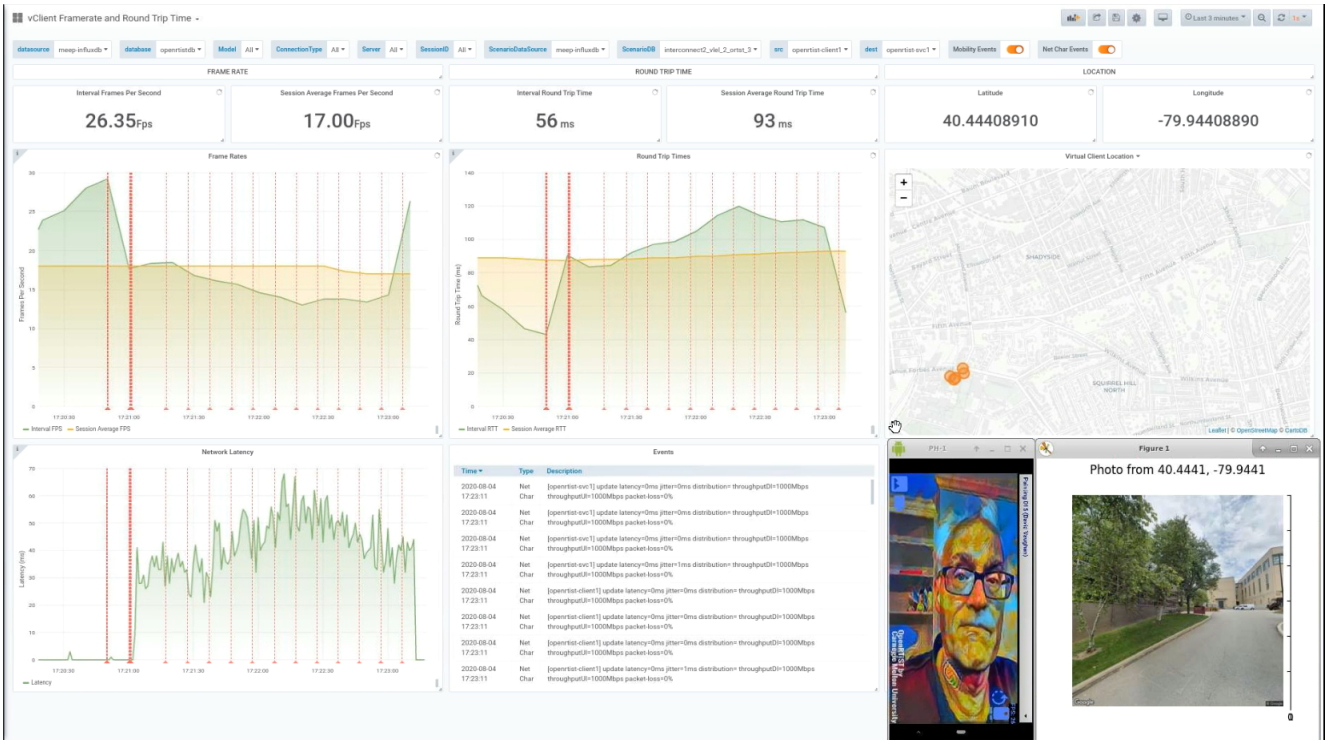

Figure 21: Walk Down Walnut Street Dashboard

4.5 Visualizing Virtual Scenarios

The other scenarios in this section focus on gathering simulation data for assessing the performance of applications in the face of varying networks. While that is the primary use of the simulation framework, we also have found a need to visualize scenarios in a way that makes the user experience during a simulation more tangible. To accomplish this, we extended the framework with GMapView [35] and Scrcpy [36] to simulate the movement of a specific UE between virtual locations while viewing the UE experience and the actual location represented by the virtual location. This visualization can best be experienced by viewing the video in [37] and the screenshot in Figure 21. The visualization includes the application RTT and FPS, the simulated network latency, a map of the virtual route, a video feed from the UE and a Google Streetview image of the current virtual location.

5 Discussion and Conclusion

This report presented a simulation framework for mobile edge computing and applied that framework to a set of simulation scenarios. The intent of this work was to show how simulation can provide insights and conclusions to real mobile edge computing challenges and to present the specific results of those scenarios. We do not present generalized conclusions from the results due to the limitations of the simulations:

- The work was completed using a single application, OpenRTIST. While OpenRTIST is an excellent representative of single user interactive applications, it does not adequately

represent all applications of all classes. Future work should expand on the representation with a particular focus on broadening to other application classes.

- The network characteristics used in the simulations were derived from real network measurements provided by our carrier partners. However, this data was limited in scope and, accordingly, simulations were run with simple scenarios. More complete simulations would include a broader sampling of real network characteristics over time and network and more complex mobility and characteristics scenarios.

- As they are for application developers, mobile networks are opaque for us. Our topology constructs are accordingly relatively simplistic when compared to real networks. Future work could benefit from access to a full real network topology and its corresponding characteristics. Fully implementing a complex topology may also require enhancement to the AdvantEDGE emulation platform to reflect the complexity.

- Our simulations reflect the typical structure and characteristics of a 4G LTE network. As networks move to 5G, the new capabilities provided by 5G will impact both the simulations and their results. For example, the introduction of mmWave technologies into networks will dramatically change the latency and bandwidth characteristics and therefore the types of applications that can achieve their minimum acceptability constraints. Future work should encompass scenarios that represent the characteristics of 5G.

- For simplicity, our simulations were run with a single UE running a single application. Real networks must support a multi-tenant configuration with many mobile UEs running diverse applications simultaneously. Many users and applications will also increase the overall network load and congestion thereby increasing the network latency, jitter and packet loss. The simulation framework can support scenarios of this type but their design and execution can be complex. For example, interactive applications usually require user interaction – a trait which is difficult to scale into hundreds of UEs. Potentially, the simulation framework could be extended to create more flexible automated UE "load generators".[24] However, that approach may lose the connection between the application user experience metrics and the simulation.

Even with these limitations, some tentative conclusions emerge:

- Edge computing requires that cloudlets are located in the same metro/region as the application users.

- Once regional cloudlets are deployed, networks and network IXPs must be engineered to avoid UE to cloudlet paths outside of the region. Since cloudlets will often be hosted on wired metro networks, regional IXPs will increase significantly in importance.

- However, once cloudlets are in the region, the marginal benefit to moving cloudlets close to the user (e.g., to the cell tower) is smaller and may not justify cost. This conclusion, however, depends on the value and requirements of the full set of edge applications to be deployed. For example, edge games such as first person shooter and driving games that rely on very fast user responses require very low and consistent RTT to be acceptable. Achieving this will necessitate moving the edge closer to the gamer. IOT Sensor applications requiring real-time or near real-time control responses will also likely need closer placements.

6 Acknowledgements

This work was conducted at the Carnegie Mellon University Living Edge Lab in Pittsburgh, Pennsylvania USA under the leadership of Professor Mahadev Satyanarayanan. The authors would like to thank Kevin Di Lallo and Robert Gazda from InterDigital for their support and guidance in the use of the AdvantEDGE platform, Simone Mangiante and Guenter Klas from Vodafone for their insights on mobile network characteristics, Alan Bock and Alex Marcham from VaporIO for their insights on carrier interconnect and Padmanabhan Pillai from Intel for his expertise in end-to-end applications. Thank you to Rolf Schuster and the Open Edge Computing initiative for direction and guidance on the simulation scenarios. Special thanks to Thomas Eiszler and Jan Harkes of the CMU Living Edge Lab for their support on infrastructure setup.

This research was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0051 and by the National Science Foundation (NSF) under grant number CNS-1518865 and the NSF Graduate Research Fellowship under grant numbers DGE1252522 and DGE1745016. Additional support was provided by Intel, Vodafone, Deutsche Telekom, Crown Castle, InterDigital, Seagate, Microsoft, VMware and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] M. Satyanarayanan. "The Emergence of Edge Computing". In: *Computer* 50.1 (2017), pp. 30–39.

[2] M. Satyanarayanan et al. "The Seminal Role of Edge-Native Applications". In: *2019 IEEE International Conference on Edge Computing (EDGE)*. 2019, pp. 33–40.

[3] Zhuo Chen et al. "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*. SEC '17. San Jose, California: Association for Computing Machinery, 2017. ISBN: 9781450350877. DOI: 10.1145/3132211.3134458. URL: https://doi.org/10.1145/3132211.3134458.

[4] M. Satyanarayanan et al. "Cloudlets: at the leading edge of mobile-cloud convergence". In: *6th International Conference on Mobile Computing, Applications and Services*. 2014, pp. 1–9.

[5] Khan Pathan and Rajkumar Buyya. "A Taxonomy and Survey of Content Delivery Networks". In: (Apr. 2012). URL: https://www.researchgate.net/publication/228620804_A_Taxonomy_and_Survey_of_Content_Delivery_Networks.

[6] Roger Pantos and William May. *HTTP Live Streaming*. RFC 8216. Aug. 2017. DOI: 10.17487/RFC8216. URL: https://rfc-editor.org/rfc/rfc8216.txt.

[7] I. Sodagar. "The MPEG-DASH Standard for Multimedia Streaming Over the Internet". In: *IEEE MultiMedia* 18.4 (2011), pp. 62–67.

[8] Kiryong Ha et al. “Towards Wearable Cognitive Assistance”. In: *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys ’14. Bretton Woods, New Hampshire, USA: Association for Computing Machinery, 2014, pp. 68–81. ISBN: 9781450327930. DOI: 10.1145/2594368.2594383. URL: https://doi.org/10.1145/2594368.2594383.

[9] Latif U. Khan et al. *Federated Learning for Edge Networks: Resource Optimization and Incentive Mechanism*. 2019. arXiv: 1911.05642 [cs.DC].

[10] S. Mehraghdam, M. Keller, and H. Karl. “Specifying and placing chains of virtual network functions”. In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. 2014, pp. 7–13.

[11] H. A. Alameddine et al. “Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing”. In: *IEEE Journal on Selected Areas in Communications* 37.3 (2019), pp. 668–682.

[12] F. Wei, S. Chen, and W. Zou. “A greedy algorithm for task offloading in mobile edge computing system”. In: *China Communications* 15.11 (2018), pp. 149–157.

[13] M. Chen and Y. Hao. “Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network”. In: *IEEE Journal on Selected Areas in Communications* 36.3 (2018), pp. 587–597.

[14] Teemu Kämäräinen et al. “A Measurement Study on Achieving Imperceptible Latency in Mobile Cloud Gaming”. In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, June 2017. DOI: 10.1145/3083187.3083191. URL: https://doi.org/10.1145%2F3083187.3083191.

[15] Shanhe Yi et al. “LAVEA: Latency-Aware Video Analytics on Edge Computing Platform”. In: *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. SEC ’17. San Jose, California: Association for Computing Machinery, 2017. ISBN: 9781450350877. DOI: 10.1145/3132211.3134459. URL: https://doi.org/10.1145/3132211.3134459.

[16] J. Martín-Pérez et al. “Modeling Mobile Edge Computing Deployments for Low Latency Multimedia Services”. In: *IEEE Transactions on Broadcasting* 65.2 (2019), pp. 464–474.

[17] C. Sonmez, A. Ozgovde, and C. Ersoy. “EdgeCloudSim: An environment for performance evaluation of Edge Computing systems”. In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 2017, pp. 39–44.

[18] Miranda McClellan. “WebRTC Based Network Performance Measurements”. MA thesis. Nov. 2019. URL: https://dspace.mit.edu/handle/1721.1/123051.

[19] V. Kartashevskiy and M. Buranova. “Analysis of Packet Jitter in Multiservice Network”. In: *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S T)*. 2018, pp. 797–802.

[20] Hamza Dahmouni, André Girard, and Brunilde Sansò. “An analytical model for jitter in IP networks”. In: *annals of telecommunications - annales des télécommunications* 67.1-2 (May 2011), pp. 81–90. DOI: 10.1007/s12243-011-0254-y. URL: https://doi.org/10.1007%2Fs12243-011-0254-y.

[21] N. Mesbahi and H. Dahmouni. “Delay and jitter analysis in LTE networks”. In: *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2016, pp. 122–126.

[22] H. Dahmouni et al. “The Impact of Jitter on Traffic Flow Optimization in Communication Networks”. In: *IEEE Transactions on Network and Service Management* 9.3 (2012), pp. 279–292.

[23] K. -. Chen, C. -. Tu, and W. -. Xiao. “OneClick: A Framework for Measuring Network Quality of Experience”. In: *IEEE INFOCOM 2009*. 2009, pp. 702–710.

[24] Manuel Osvaldo J. Olguın Muñoz et al. “EdgeDroid: An Experimental Approach to Benchmarking Human-in-the-Loop Applications”. In: *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. HotMobile ’19. Santa Cruz, CA, USA: Association for Computing Machinery, 2019, pp. 93–98. ISBN: 9781450362733. DOI: 10.1145/3301293.3302353. URL: https://doi.org/10.1145/3301293.3302353.

[25] Carnegie Mellon University. *The Living Edge Lab*. https://openedgecomputing.org/living-edge-lab/. 2020.

[26] Michel Roy, Kevin Di Lallo, and Robert Gazda. *AdvantEDGE: A Mobile Edge Emulation Platform (MEEP)*. https://github.com/InterDigitalInc/AdvantEDGE. 2020.

[27] S. George et al. “OpenRTiST: End-to-End Benchmarking for Edge Computing”. In: *IEEE Pervasive Computing* (2020), pp. 1–9. DOI: 10.1109/MPRV.2020.3028781.

[28] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “Image Style Transfer Using Convolutional Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[29] *OpenAPI Specification – Version 2.0*. http://spec.openapis.org/oas/v2.0. 2014.

[30] *Swagger Codegen Website*. https://swagger.io/tools/swagger-codegen/. 2020.

[31] Network Encyclopedia. *Nagle’s algorithm*. URL: https://networkencyclopedia.com/nagles-algorithm/.

[32] *The Open Edge Computing Initiative*. https://openedgecomputing.org/. 2020.

[33] Tomasz Gerszberg. *Shared Edge Experience*. https://www.linkedin.com/pulse/shared-edge-experience-tomasz-gerszberg/. 2019.

[34] Unwired Labs. *The World’s Largest Open Database of Cell Towers*. https://opencellid.org/. 2020.

[35] James Blakley. *GMapView: A Python Library for Routes and StreetView images*. https://github.com/cmusatyalab/GMapView. 2020.

[36] *Scrcpy*. https://github.com/Genymobile/scrcpy. 2020.

[37] *A Walk Down Walnut Street*. https://youtu.be/OW1J-J2nWMQ. 2020.