MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing

Márcio Moraes Lopes
Institute of Computing, University of Campinas
Campinas, Sao Paulo, Brazil
marcio@lrc.ic.unicamp.br

Wilson A. Higashino
Department of Electrical and Computer Engineering, Western University
London, Ontario, Canada
whigashi@uwo.ca

Miriam A. M. Capretz
Department of Electrical and Computer Engineering, Western University
London, Ontario, Canada
mcapretz@uwo.ca

Luiz Fernando Bittencourt
Institute of Computing, University of Campinas
Campinas, Sao Paulo, Brazil
bit@ic.unicamp.br

ABSTRACT

Low latency in IT applications is an important aspect of improving the quality of the user's experience. Frequently, applications are run in a virtual machine in the cloud. Because cloud providers are datacentre facilities that are often distant from users, unacceptably high latencies are experienced in some applications. Fog computing can be seen as a cloud computing extension, namely cloudlets, located in access points at the edge of the network and hence able to provide lower latencies than the cloud. However, as mobile devices and applications become more popular, users' computing and data capacities should be maintained close to the user to keep latencies as low as possible. This paper discusses resource allocation in fog computing in the face of users' mobility and introduces MyiFogSim, an extension of iFogSim to support mobility through migration of virtual machines between cloudlets. Moreover, a migration policy is proposed, and MyiFogSim is used to analyze the policy impact on application quality of service. Results suggest that the policy can promote lower latencies when compared to a scenario without the migration policy.

CCS CONCEPTS

• **Computing methodologies → Planning and scheduling**; *Distributed computing methodologies*; • **Computer systems organization → Cloud computing**; *Distributed architectures*;

KEYWORDS

Fog Computing; Virtual Machine Migration; Edge Computing; Cloud Computing

UCC '17 Companion, December 5–8, 2017, Austin, TX, USA.
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5195-9/17/12…$15.00
https://doi.org/10.1145/3147234.3148101

1 INTRODUCTION

Fog computing is an emerging paradigm that extends cloud computing services to the edge of the network to offer low latency through geographical distribution, in contrast to the more centralized cloud [4]. In addition, fog computing can provide faster access for final users than cloud computing, but cloud computing is still more suitable than fog computing for massive data processing. Therefore, the edge capacity furnished by the so-called cloudlets – computing capacity at the user's access points – acts together with the cloud to cover a wider set of applications and their needs.

With the Internet of Things (IoT), many devices can act as sensors and have some processing power. However, the computing capacities of these devices are limited. Cloudlets provided by fog computing can be used to process IoT data, both improving latencies and reducing network traffic to the cloud. IoT devices can be mobile, including smart devices with user applications. As users and devices move from one access point to another, data and processing related to each user's device(s) and application(s) move also. This study has used virtual machines (VMs) as the mechanism to hold users' applications and data. This is motivated by the migration capabilities of VMs, which can be used to move applications and data through cloudlets according to device mobility.

Although VM migration in cloud computing is important to support [1] load balancing, power efficiency, fault tolerance, and system maintenance, on the other hand, VM migration in fog computing is also important to support user mobility. With cloudlets scattered over an urban area, for example, a VM can be migrated to cloudlets according to user position, resulting in lower latencies and better quality of experience (QoE) than in scenarios with no migration.

This paper describes MyiFogSim, which extends the iFogSim [7] simulator to support virtual machine migration policies for mobile users. Moreover, a VM migration policy for mobile users in fog computing is introduced. Results that validate the MyiFogSim implementation are shown, and a VM migration policy is evaluated against a fog computing scenario with no VM migration.

This paper is organized as follows. Section 2 discusses related work. Section 3 presents basic concepts, and Section 4 introduces the proposed migration techniques. Section 5 presents the simulators used and the proposed extension, and Section 6 presents simulation results. Section 7 concludes the paper.

2 RELATED WORK

Taleb et. al. [9] proposed Follow Me Edge (FME), which is an approach based on services hosted by the nearest cloudlet (edge) to users within smart cities scenarios. They indicated that these services should follow users. Their research was focused on MEC (Mobile Edge Computing) autonomous service creation to provide users access from anywhere and anytime, optimizing Quality of Experience (QoE) and reducing latency. The authors presented two case studies to exemplify the requirements that FME can support. FME assumes the use of light virtualization technology and also introduces concerns related to container live migration that were focused on mobility aspects in the edge.

Virtualization aspects of fog computing have been researched by Osanaiye et. al. [8] together with security and privacy issues and available services and resources. Furthermore, they reviewed VM migration in fog computing and presented a conceptual framework for a smart pre-copy VMs in a live migration approach. The authors constructed a taxonomy to distinguish two kinds of fog computing applications: i) real-time applications and ii) near real-time applications.

Farris et. al. [6] revealed several conflicting aspects of delay-sensitive applications. The authors mentioned that to have an ultra-short response time in mobile user applications, quick reallocation of resources between edge nodes must be considered. In addition, they proposed that proactive service replication is a promising strategy to reduce application downtime and guarantee an acceptable QoE. However, the placement of VM replicas in several nodes increases resource consumption, and this cost turns out to be a relevant aspect.

Yao et. al [10] introduced the Road Side Cloudlet (RSC) with the aim to resolve the low response-time problem and the high communication cost of the Vehicular Cloud Computing (VCC) model proposed in [11]. The authors of [10] considered that this approach can have a very high execution cost when VCC is used, particularly in long-distance networks. Therefore, the authors of [10] proposed a two-phase heuristic to minimize VM migration cost among RSCs during vehicle movement. Phase One is an algorithm to find the shortest path to the next node without violating link capacity constraints and Phase Two is another algorithm that is based on VM allocation cost in an RSC because the results of this phase are based on one execution.

3 BASIC CONCEPTS

The popularization of electronic devices has not only made computing capacity available anywhere (i.e., ubiquitous computing), but also data generation capacity scattered through a myriad of devices and sensors, culminating in the so-called Internet of Things (IoT). Nowadays, data from IoT devices are often sent to be stored and processed in the cloud. With the increasing number of devices, the amount of data to be transferred and processed is expected to increase significantly.

Fog computing emerged to mitigate problems that arise when centralized clouds are used to handle large amounts of data generated at the edges [4]. By introducing cloudlets closer to the edge (e.g., at the access points), fogs can pre-process (e.g., filter, aggregate, or first-pass analyze) data and send results back to devices (e.g., smart devices or actuators) without the need for long-distance communication. This approach can reduce latencies experienced by edge devices and also reduce traffic in the network links towards cloud datacentres [2].

When devices are mobile, the increasing distance to offloaded applications data or processing capacity will also increase latencies and global network traffic. In this scenario, resource management, through resource allocation decision-making, plays a key role in maintaining both latencies and data traffic at the lowest possible levels. A fog computing infrastructure should be able to keep data and processing as close as possible to their owners and/or consumers. This paper states that this can be accomplished using virtual machine migration, as discussed next.

3.1 Virtualization in fog computing

To support migration of users' content, it is assumed that the fog architecture includes capabilities to handle a virtualized environment that can migrate each user's context among cloudlets. An example of such an architecture to support virtual machine migration in fog computing is presented in [3]. Each user has a virtual machine (or a set of virtual machines) that contain his/her data and application modules/services. The key aspect of a resource management framework in this scenario is to keep virtual machines as close as possible to their owners, resulting in a resource allocation optimization problem. With user mobility, keeping VMs at (ideally) one hop distance from their owners relies on VM migration decision-making based on each user's current location, movement direction, and speed. Other factors that may play an important role in decision-making are the amount of data that each user has (i.e., the size of a VM) and historical mobility patterns. Taking these factors into account, the migration should pursue the objective of reducing latencies and downtime for accessing VMs. By combining those factors and objectives, a VM migration policy can be implemented.

3.2 Illustrative scenario

Several scenarios can be constructed when combining virtual machine migration with user mobility and mobile device network handoff. Such scenarios have certain aspects in common:

(1) Cloudlets and their geographical location.
(2) Networks: (i) between user and access point; and (ii) between access point and cloudlet running the user VM.
(3) Mobility timeline: direction and speed.

The mobile user changes his/her access points through a handoff mechanism, which is responsible for disconnecting the user from one access point and connecting to another. VM migration is a parallel task: it can start and end anytime, independently of the handoff process. Therefore, when a handoff occurs and a VM migration does not, the network path between user and VM becomes different, and, with potentially more network links between the user and a VM, higher latency is expected. Note that it is outside the scope of this paper to discuss how handoff decisions and VM

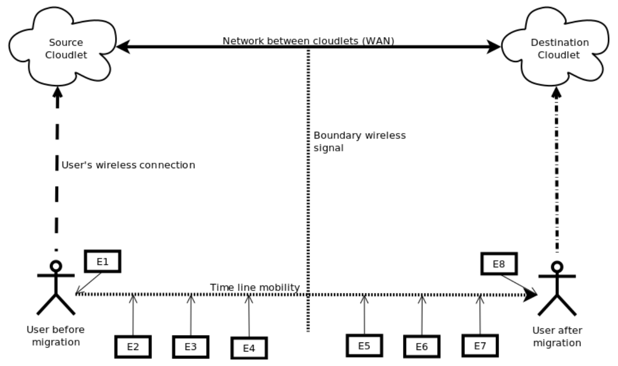

Figure 1: Illustrative scenario of migration events.

migration actually occur: this paper focuses on the decision on when to start the migration and where to migrate a VM.

Figure 1 shows an illustrative handoff and migration scenario with user mobility in fog computing, as detailed below.

Cloudlets: The *source cloudlet* is running the user's VM before the migration and handoff processes. The user is connected through a wireless connection link.

The *destination cloudlet* will run the user's VM after the migration and handoff processes. The ideal migration should guarantee that when the user crosses the "handoff zone" (the wireless signal boundary), the VM should already be placed in the destination cloudlet. As mentioned before, the VM size can impact how long the migration takes, and therefore it can also impact how long before the user reaches the handoff zone the migration should be started.

Network and Wireless Connection: The *Network between cloudlets (WAN)* is the physical network that connects cloudlets among each other. This network can present different topologies and several network types, for example optical fiber, wireless, satellite, and PLC (Power Line Communications).

A *User's wireless connection* is a wireless network between the user's mobile device and the access point, as for example 4G/LTE, 5G, Wi-Fi, Bluetooth, and so on. In this scenario, the user is connected to the source cloudlet (before the migration process) and to the destination cloudlet (after the migration process).

In this example scenario, eight events can be seen in the mobility timeline.

E1 *Decision to perform migration*: The time instant at which the migration algorithm decides that a migration is needed and chooses the destination cloudlet.

E2 *Preparing the migration*: Identify migration needs (e.g., amount of data) and establish a network connection between the source and destination cloudlets.

E3 *Migration process start*: Start migrating the virtual machine, i.e., sending data from the source cloudlet to the destination cloudlet.

E4 *Handoff start*: This event is defined by the network policy that handles handoff. In an ideal scenario, it should occur after the start of the migration process because the whole handoff process is quicker than the migration of a VM.

E5 *Handoff end*: At this time, the handoff mechanism is completed. The user is now connected to the access point closer to the destination cloudlet, waiting for migration to finish.

E6 *Migration process end*: At this time, the virtual machine migration process is completed. The virtual machine should be ready to receive user requests.

E7 *Cloudlet request*: The user sends a request to the new cloudlet and starts using the virtual machine resources again.

E8 *Virtual machine access*: The user can normally access applications and data in the new cloudlet.

The steps illustrated above may occur in a different order, which can impact VMs downtime during migration or the average latencies experienced by the user. Two scenarios with different event orders are discussed below.

3.2.1 *Delayed migration decision.*

If a migration decision is delayed, the handoff process may start before the migration decision takes place. In this case, the user will access his/her VM from a new access point, but the VM will still be in the source cloudlet. Although VM migration does not take place, the latencies observed by the user will be a function of the underlying network connecting the source and destination cloudlets. In this scenario, the time sequence of events would be: E1, E2, E4, E5, E3, E6, E7, and E8.

3.2.2 *Concomitant decision.*

An ideal scenario for the user occurs when the handoff and VM migration start at the same time. If both events also end together, VM downtime can be reduced. In this case, there is no wait between the time when the connection to the new access point has been established and the use of the VM (i.e., between E5 and E6). In this scenario, the time sequence of events would be: E1, E2, E3/E4, E5/E6, E7, and E8.

4 MIGRATION TECHNIQUE

Based on the architecture described in the previous section, a new migration technique has been designed. This new technique is composed of two main parts:

- **Migration policy**: this defines *when* a user VM should be migrated, considering aspects such as the user's speed, direction and geographical position.
- **Migration strategy**: this determines *where* the user's VM is going, i.e., the destination cloudlet, and *how* the migration is performed.

4.1 Migration policy

The migration policy examines when the user VM should be migrating according to the physical aspects of the scenario. Figure 2 illustrates the elements that must be monitored to implement the migration policy in a simple topology composed of a single access point (AP).

The **user** has a position, speed, and direction that define his/her movement. The migration policy verifies these attributes relative to the AP positions and decides, at each verification point, whether the user VM should migrate.

VM migration is considered whenever a user is located in his/her **migration zone**, which is the geographical conical region defined by the two direction edges adjacent to the user's direction. For instance, the *user 1* migration zone is the region between the northeast and southeast edges because he/she is moving to the east. Similarly, the *user 2* migration zone is defined by the east and south edges.

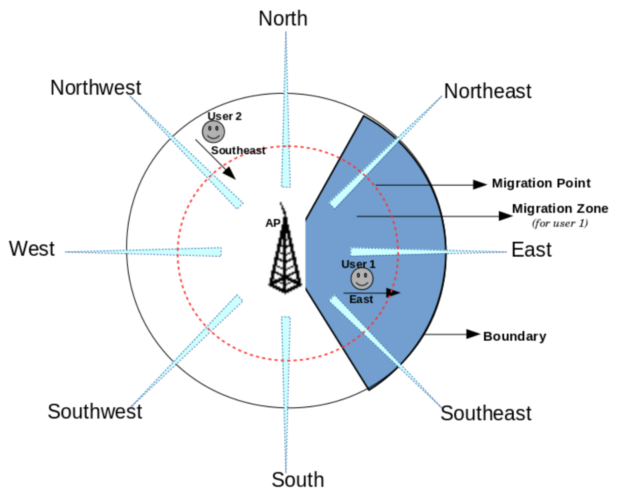

Figure 2: Point to make the migration decision.

The **migration point** is the point within the migration zone at which the migration starts. In this work, this point is determined based on a fixed distance to the AP or on dynamic distances based on the user's speed and VM size. Note that, in both cases, the migration point is located before the network handoff position so as to guarantee the best qualify of experience to the users.

All these components are parameters used for the migration decision and are also associated with the migration strategy, which is described next.

4.2 Migration strategy

The *migration strategy* defines two important aspects of the VM migration: *where* the VM should be migrated, and *how* this migration should be implemented.

This research explored three different strategies to determine the destination *cloudlet*:

- **Shortest distance between device and AP**: this strategy selects the *cloudlet* connected to the next AP closest to the user;
- **Shortest distance between device and cloudlet**: this strategy chooses the next cloudlet closest to the user.
- **Lowest Latency**: this strategy selects the cloudlet with the lowest latency among a set of cloudlet candidates.

Furthermore, three techniques were explored to implement VM migration:

- **Full VM migration (non-live)**: based on traditional VM migration, this technique creates an image of the entire VM and delivers it to the transport layer.
- **Container migration (non-live)**: based on container migration, this technique reduces the amount of data to be transported but it introduces additional complexity due to creation of the container image.
- **Live migration**: based on live-migration post copy, this technique delivers the VM data in two steps: 1) system and user data that are not in main memory; 2) data in main memory.

5 SIMULATOR OVERVIEW

5.1 CloudSim and iFogSim

Calheiros et al. [5] implemented Cloudsim, a general and extensible framework that simulates cloud computing infrastructure and application services. The authors mentioned that researchers and developers can use CloudSim to test the performance of services and, if necessary, fix or improve the system.

Based on CloudSim, iFogSim is a toolkit that has been developed by Gupta et al. [7] in 2016. iFogSim supports fog computing and IoT concepts and can be used to measure the impact of resource management techniques from latency, network utilization, power consumption, and cost perspectives [7].

Although iFogSim already implements important fog computing components, the following features needed to simulate user VM migration were still missing:

- User device mobility, including positioning on a map;
- Radio base station (Access Point);
- Migration of virtual machines;
- Migration strategies and migration policies.

This research proposes MyiFogSim [1], an extension of iFogSim that aims to overcome these limitations.

5.2 MyiFogSim

Figure 3 shows a class diagram of the MyiFogSim implementation. Classes shown in dark color were provided by CloudSim and iFogSim, whereas the others are MyiFogSim classes that have been incorporated into the original design.

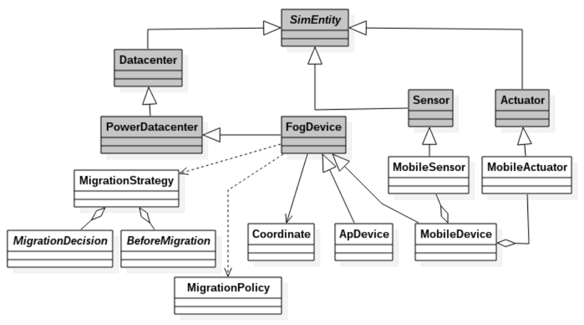

Figure 3: Core of MyiFogSim architecture.

These classes are defined as follows:

- **Coordinate:** This class represents map coordinates (x,y) in a Cartesian plane and is used to designate the placement of all simulated entities. The map boundaries can be defined by MyiFogSim users.
- **ApDevice:** This class extends FogDevice from iFogSim and adds the features and responsibilities of a wireless network access point, such as a handoff management system and connection/disconnection of devices.

[1] Available at http://www.lrc.ic.unicamp.br/fogcomputing/

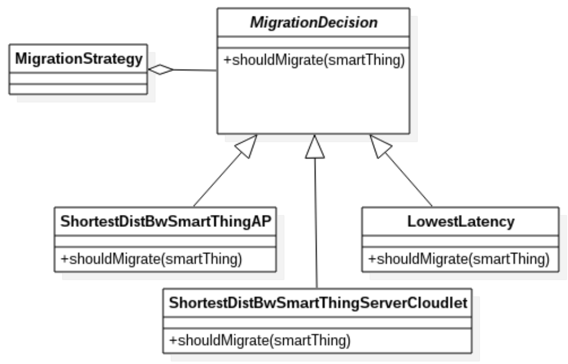
MigrationStrategy
MigrationDecision
+shouldMigrate(smartThing)
ShortestDistBwSmartThingAP
+shouldMigrate(smartThing)
LowestLatency
+shouldMigrate(smartThing)
ShortestDistBwSmartThingServerCloudlet
+shouldMigrate(smartThing)

Figure 4: Design pattern strategy for migration decision.

- **MobileDevice:** This class also extends FogDevice from iFogSim. Its main goal is to separate the concerns of fog servers and user devices because in iFogSim both of these are represented by instances of the *FogDevice* class. With this separation, it is possible to keep features particular to user or IoT devices in their own class.
- **MobileSensor:** This class extends Sensor from iFogSim and represents a set of sensors that exists within a user device. Conversely, in iFogSim, the developer must create several Sensor objects to model hardware with multiple sensors.
- **MobileActuator:** This class extends Actuator from iFogSim and provides abstractions similar to MobileSensor, but representing actuators.
- **MigrationStrategy:** This class has been designed following the design pattern methodology and represents a migration strategy. A researcher/developer can design different migration strategies by creating implementations of this class.
- **MigrationPolicy:** Similarly to MigrationStrategy, this class follows the design pattern strategy and can be used to implement the migration policies proposed by the researcher/developer.

MyiFogSim implements migration of AppModules, which represent the application components, similarly to how CloudSim handles VM migration. A method in the FogDevice class is responsible for:

(1) disabling all mappings between tuples, application modules, user device, and the source cloudlet;
(2) enabling again the new mapping in the destination cloudlet.

Furthermore, note that during a migration or a handoff process new tuples are built by sensors, but they are not processed by the user device or the source cloudlet. Therefore, if an application requires all tuples to be processed, a buffer should be built to accumulate tuples and re-enable their processing after the migration or handoff.

5.2.1 Migration Decision Interface. Figure 4 shows a detailed class diagram of the MigrationDecision interface. The interface has just one boolean method named shouldMigrate that must be implemented according to the policy chosen. MyiFogSim contains built-in implementations for all policies described in Section 4.2.

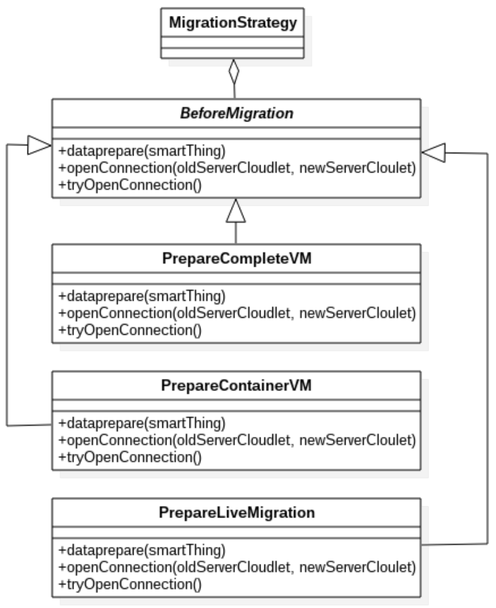

Figure 5: Design pattern strategy before migration.

5.2.2 Before Migration Interface. Figure 5 shows a class diagram of the BeforeMigration interface. This interface has three methods: i) dataprepare; ii) openConnection; iii) tryOpenConnection. The dataprepare method is the main interface method and must be implemented according to the migration policy and migration strategy. MyiFogSim provides built-in implementations for the three methods used to prepare data for migration described in Section 4.2.

6 RESULTS

Simulations were performed to validate and assess virtual machine migration using MyiFogSim. The *EEG Tractor Beam Game* was used as implemented in iFogSim [7], which is a concentration game where users can attract virtual objects towards themselves using electroencephalography. The simulations were divided into two parts. First, as a baseline, a single user travelled across the map, performing handoffs when necessary, but without any virtual machine migration. In this scenario, a VM was running in the first cloudlet where the user's device had been connected. In the second part, the simulation was similar, but VM migrations were performed according to the chosen policy and strategy.

To perform a preliminary evaluation, a simplified mobility model was used, where a user started at a corner of the map and moved towards the most distant, opposite corner in a 40×40 kilometre map. The users speed was set to 19 m/s (~ 70 km/h) with no changes in direction. Cloudlets and access points were uniformly distributed over the map at intervals of 10 and 6.5 kilometres respectively. Access point antennas were set up with a 5-kilometre transmission radius. A simple handoff policy was implemented in which the handoff occurred at a fixed point 40 metres before the user reached the transmission boundary of the antenna where he/she was currently connected. The static migration point was set up to occur 30% (in metres) before the handoff process and VMs had a fixed size of 186 MB. The bandwidth was defined as 1 Mbps (uplink) and 2 Mbps (downlink) between user and access point and uniformly distributed between 10 and 20 Mbps between cloudlets. Link delays were introduced by the NetworkTopology.addLink() method from CloudSim.

To illustrate latency behavior in the fog, Figure 6 shows the latency resulting from the two scenarios – with and without VM migration – for a single simulation. The beginning of the latency curves in both scenarios were the same because the user was accessing his/her VM in the first cloudlet that was connected. After 200 seconds of simulation, the no migration curve increased latency from 20 ms to about 125 ms. This was due to a handoff to a new access point, but without migrating the user's VM, which remained in the previous cloudlet. Similar behavior occurs at 800, ~950, ~1300, ~1450 seconds. When VM migration occurred, the delay remained around 20 ms because the VM was migrated along with the user.

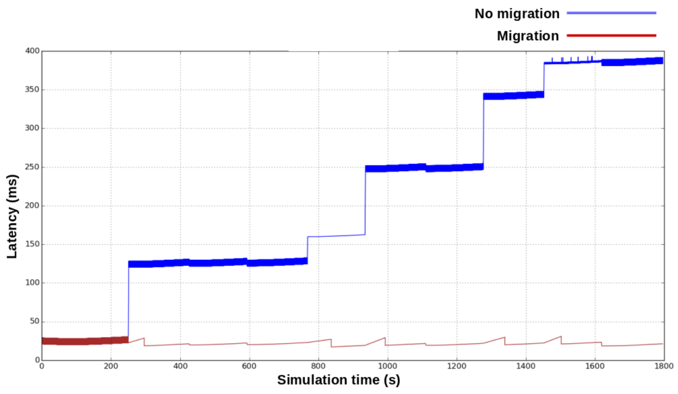

Figure 6: Latencies with and without VM migration.

Figure 7 shows latencies observed during the simulations. The observed latency was greater than 200 ms without migration, which can impair application execution depending on the quality of service needed. On the other hand, migration strategy 3 showed better performance in this preliminary simulation.

7 CONCLUSION

Users and their mobile devices have posed great challenges to researchers and technology service providers in recent years. Fog computing establishes an infrastructure that can address many of these challenges.

This paper has presented an overview architecture for virtual machine migration in fog computing with the aim of maintaining mobile users' quality of experience. MyiFogSim was implemented to simulate scenarios with mobile users, and preliminary simulation results showed that supporting VM migration and introducing migration policies into a fog computing infrastructure can improve quality of service through reduced latencies because VMs can be maintained closer to the mobile users. Future work will include the development of more advanced policies that can incorporate user behavior and applications requirements.

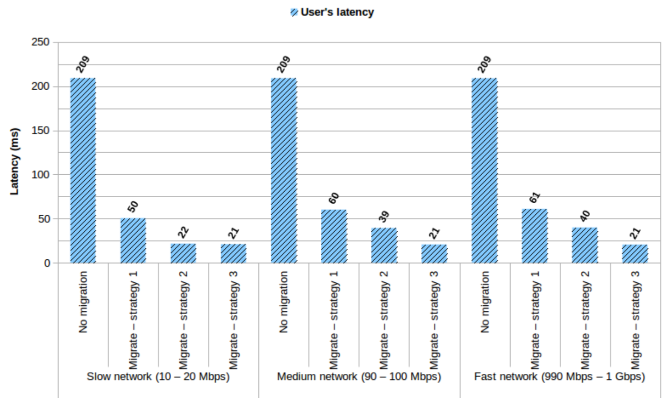

Figure 7: Latencies for different migration strategies and network throughput.

ACKNOWLEDGMENTS

This work was partially funded by CAPES, CNPq, and the European Commission H2020 programme (grant 688941 – FUTEBOL), as well by the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through RNP and CTIC. We also would like to thank the Global Affairs Canada for financial support through ELAP 2016/2017.

REFERENCES

[1] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Shiraz, Abdullah Yousafzai, and Feng Xia. 2015. A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *Journal of Network and Computer Applications* 52 (2015), 11–25.

[2] Luiz F. Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F. Rana, and Manish Parashar. 2017. Mobility-Aware Application Scheduling in Fog Computing. *IEEE Cloud Computing* 4, 2 (March 2017), 26–35. https://doi.org/10.1109/MCC.2017.27

[3] Luiz F. Bittencourt, Márcio Moraes Lopes, Ioan Petri, and Omer F. Rana. 2015. Towards virtual machine migration in fog computing. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on.* IEEE, 1–8.

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12).* ACM, New York, NY, USA, 13–16. https://doi.org/10.1145/2342509.2342513

[5] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

[6] Ivan Farris, Tarik Taleb, Miloud Bagaa, and H Flinck. 2017. Optimizing service replication for mobile delay-sensitive applications in 5g edge network. *IEEE* 201 (2017), 2017.

[7] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. *arXiv preprint arXiv:1606.02007* (2016).

[8] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. Choo, and M. Dlodlo. 2017. From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access* PP, 99 (2017), 1–1. https://doi.org/10.1109/ACCESS.2017.2692960

[9] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck. 2017. Mobile Edge Computing Potential in Making Cities Smarter. *IEEE Communications Magazine* 55, 3 (March 2017), 38–43. https://doi.org/10.1109/MCOM.2017.1600249CM

[10] Hong Yao, Changmin Bai, Deze Zeng, Qingzhong Liang, and Yuanyuan Fan. 2015. Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud. *Concurrency and Computation: Practice and Experience* 27, 18 (2015), 5780–5792.

[11] Rong Yu, Yan Zhang, Stein Gjessing, Wenlong Xia, and Kun Yang. 2013. Toward cloud-based vehicular networks with efficient resource management. *IEEE Network* 27, 5 (2013), 48–55.