Dynamic Network Slicing in Fog Computing for Mobile Users in MobFogSim

Diogo Gonçalves*, Carlo Puliafito†, Enzo Mingozzi†, Omer Rana‡, Luiz Bittencourt*, and Edmundo Madeira*
*Institute of Computing, University of Campinas, Brazil
†Department of Information Engineering, University of Pisa, Italy
‡School of Computer Science and Informatics, Cardiff University, UK
diogomg@lrc.ic.unicamp.br, carlo.puliafito@ing.unipi.it,
enzo.mingozzi@unipi.it, ranaof@cardiff.ac.uk, {bit, edmundo}@ic.unicamp.br

***Abstract*—Fog computing provides resources and services in proximity to users. To achieve latency and throughput requirements of mobile users, it may be useful to migrate fog services in accordance with user movement – a scenario referred to as *follow me cloud*. The frequency of migration can be adapted based on the mobility pattern of a user. In such a scenario, the fog computing infrastructure should simultaneously accommodate users with different characteristics, both in terms of mobility (e.g., route and speed) and Quality of Service requirements (e.g., latency, throughput, and reliability). Migration performance may be improved by leveraging "network slicing", a capability available in Software Defined Networks with Network Function Virtualisation. In this work, we describe how we extended our simulator, called MobFogSim, to support dynamic network slicing and describe how MobFogSim can be used for capacity planning and service management for such mobile fog services. Moreover, we report an experimental evaluation of how dynamic network slicing impacts on container migration to support mobile users in a fog environment. Results show that dynamic network slicing can improve resource utilisation and migration performance in the fog.**

***Index Terms*—Network Slicing, Fog Computing, Internet of Things, Mobility, Container Migration, Follow Me Cloud, Simulator, MobFogSim.**

I. Introduction

The Internet of Things (IoT) is a popular paradigm for the implementation of electronic services and applications. However, IoT devices can have computational, storage, and battery constraints and require a cloud data centre for the execution of compute-intensive tasks. Cloud servers are generally concentrated in a few data centres and are typically multiple hops from IoT devices, making the cloud data centre unable to meet latency constraints of many emerging applications.

In this context, *fog computing* provides computing resources closer to the network edge, often in close proximity to end users and devices needing them. As fog computing comprises a number of distributed nodes – rather than a single (often centralised) data centre, it can offer benefits such as: (i) low latency; (ii) better privacy and security; (iii) improved context awareness; and (iv) uninterrupted services in the presence of intermittent or limited network connectivity to the cloud [1]. However, fog computing does not replace the requirement for a cloud data centre, but rather complements it, providing a hierarchical infrastructure between the network edge and a centralised cloud data centre. Fog computing resources are synonymously referred to as fog nodes, cloudlets, or Micro Data Centres (MDCs) and usually run services in the form of Virtual Machines (VMs) or containers.

The implementation of fog computing presents several challenges. Device mobility is one of them because, when a device/user moves, the communication distance to the fog service may increase, and benefits of using fog nodes may no longer be possible. A possible approach to solve this issue is to migrate the user's VM/container across the fog infrastructure to keep it always *close enough* to the mobile user [2].

These *service migration* and resource management approaches are made more complex due to significant diversity and heterogeneity of users and application requirements that exist in fog environments [3] [4]. Mobile users may move at different speeds (e.g., on foot or by car) and according to different mobility patterns. In addition, some applications may present strict requirements in terms of latency; others may prioritise throughput, while others may have specific hardware needs. In such a scenario, some users may require a higher priority for managing their service migration, manage their migration speed, or require specific destination node(s). Traditional network infrastructures are not able to guarantee all of these, sometimes parallel, levels of security, performance, reliability, privacy, and/or other particular requirements for each user group.

Software Defined Networking (SDN), Network Functions Virtualisation (NFV), and dynamic network slicing offer mechanisms to support some of these requirements. Specifically, SDN and NFV [5] decouple network functions (e.g., firewalls, routing) from the underlying network hardware, implementing them as Virtual Functions. This leads to great flexibility, as it enables the creation, deletion, and migration of network functions from one hardware device to another on-demand and in a timely way. Network slicing [6] leverages SDN and NFV to create different logical/virtual networks on top of the same physical network. Each logical network (i.e., a network slice) is tailored to fulfill the requirements of a particular group of users and/or applications. Furthermore, network slices can be dynamically reconfigured and reallocated to address the changing needs of users and avoid underutilisation of network resources (i.e. via dynamic network slicing).

In [7] we introduced *MobFogSim* – a simulator for VM/container migration in fog environments serving mobile users. In this work, we extend MobFogSim to support dynamic network slicing and analyse the impact of dynamic network slicing on migration performance in the fog. Specifically, the contributions of this paper are twofold:

- it discusses our design and implementation of dynamic network slicing in MobFogSim. Dynamic network slicing represents an important extension to this simulator, which currently is the only fog computing simulator that implements VM/container migration to support mobile users;
- it evaluates the impact of dynamic network slicing on container migration in the fog to support mobile users. We present our experimental contributions in two parts. The first analyses static allocation of network slices, and the second considers dynamic slices. Simulations were carried out in MobFogSim.

The rest of the paper is organised as follows. Section II introduces key concepts and context for this work. Section III introduces the most salient aspects of MobFogSim and describes experiments that motivated this work. Section IV presents the main modifications that we made in MobFogSim to add dynamic network slicing support. Sections V and VI describe the experiment setup and provides an analysis of results – for both static and dynamic network slicing. Section VII outlines open challenges that need consideration, and related work is presented in Section VIII. Finally, Section IX concludes this paper and presents future works.

II. Background

This section provides an overview of the core concepts in this work: container migration (see Section II-A) and network slicing (see Section II-B).

A. Container migration

Fog computing applications are typically encapsulated within virtual environments, such as VMs or containers, which allow multi-tenancy and high flexibility. MobFogSim models both VMs and containers (see Section III-A). However, we highlight that even though the choice between VMs and containers depends on actual user needs and application requirements, the latter are more lightweight than the former in terms of memory consumption and therefore in terms of data to transmit during migration. Hence, *containers* are typically preferred in fog computing environments [8], where resources are more limited than in cloud data centres. Based on this, we mainly consider containers in our work. However, the key ideas outlined in this work are also applicable to VMs.

Different techniques exist to migrate containers. The four most popular approaches are: cold, post-copy, pre-copy and hybrid. Currently, MobFogSim models the first two of these approaches. For this reason, we now provide a brief description of these two only. However, the work in [9] provides a thorough discussion and quantitative comparisons for these four techniques.

Cold migration stops the container before saving and transmitting its state; it then resumes the container at the destination only when all the state has been transferred. Therefore, the time interval during which the container is unavailable to its users (i.e., *downtime*) coincides with the *total migration time*, which is the overall time required to complete the migration.

Post-copy migration involves stopping the container only for saving and transferring a small portion of the overall state. This leads to a more limited downtime. The container then resumes at destination, and, while it is running, the rest of its state gets migrated. Post-copy is a "live" migration technique, because the container remains active most of the time during the migration process.

B. Network slicing

In traditional networks, network functions are embedded in the hardware. Under such conditions, changes in demand results in physical changes to the network infrastructure. There are situations in which application requirements cannot be always fulfilled by this kind of infrastructure. One example is that of mobile users who present different characteristics of mobility (e.g. speed of movement) and have different Quality of Service (QoS) requirements (e.g., latency, throughput). NFV decouples network functions from the underlying hardware and presents them as virtual functions. SDN, instead, introduces the concept of programmable networks in which it is possible to control network resources through Application Programming Interfaces (APIs). The combination of NFV and SDN enables the concept of network slicing.

Network slicing creates different logical networks over the same physical network. Each logical network (i.e., a network slice) is created to serve a group of users with specific needs. Network slices can cover all the network domains (i.e., end-to-end slice) or only some of them. Furthermore, slices can present different levels of isolation: some of the logical resources or network functions of a slice can be shared with other slices or actors outside the domain. Network slices enable better control of network resources and provide users with a virtual network that is tailored to their requirements and characteristics.

Fig. 1 illustrates the concept of network slicing. As shown, there is a physical network infrastructure on top of which there are three logical networks. Each of them addresses the requirements of a specific category of applications. Hence, the first logical network is dedicated to enhanced Mobile BroadBand (eMBB) applications, which require high throughput. The second slice supports Machine-type Communications (mMTCs), which requires efficient connections in terms of energy consumption in an environment with high connection density. The third and last logical network is meant to serve Ultra-Reliable Low-Latency Communications (URLLCs) applications, which require high reliability and availability with ultra-low latencies.

Network resources can be assigned to each slice either statically or dynamically. A *static network slice* receives a fixed portion of network resources according to its demand. This

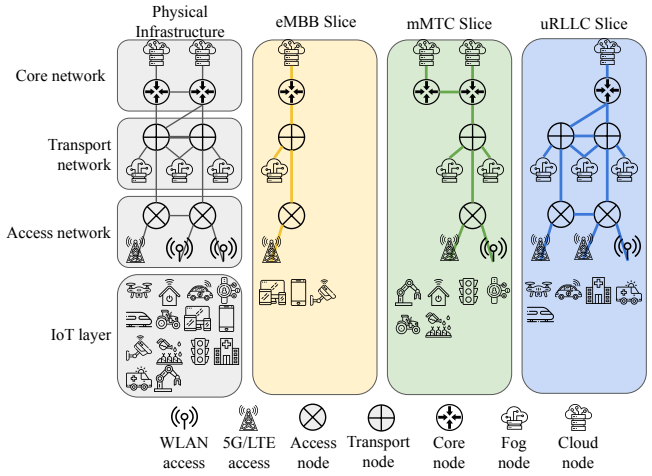


Fig. 1: Types of Network Slicing

number of resources remains allocated to that slice until the slice is active. Further resources cannot be assigned to a static slice at runtime. On the contrary, *dynamic network slicing* allows a slice to acquire and release resources according to dynamic demand, thus enabling scaling up and scaling down.

III. MobFogSim

Section III-A presents MobFogSim and provides an overview of its mobility and migration modelling. In Section III-B we discuss some preliminary simulations in MobFogSim that motivate this work.

A. Overview of MobFogSim

MobFogSim [7], derived from MyiFogSim [10], is an open-source fog computing simulator[1] developed as an extension of iFogSim [11]. The latter is written in Java and is widely used for modelling and simulating fog computing networks. Applications in iFogSim are modelled as Directed Acyclic Graphs (DAGs) where vertices represent application modules, while edges represent interactions expressed as offloading of tasks (called "tuples" in iFogSim) between modules. In terms of physical infrastructure, iFogSim models fixed IoT devices, fog nodes and cloud data centres.

MobFogSim enhances iFogSim with the modelling of user mobility and VM/container migration among fog nodes. This makes MobFogSim a comprehensive tool for the evaluation of fog computing networks where end user devices are both fixed and mobile. More specifically, MobFogSim implements a *Migration Policy* that decides when to migrate a user's VM/container. This decision is based on parameters such as: (i) user's position, direction, and speed; and (ii) occurrence of a wireless handoff (i.e., change of access point performed by the mobile device). The *Migration Strategy*, instead, decides where to migrate the service, i.e. it chooses the next fog node to run the service. This selection may be based on geographical or topological proximity of the new fog node to the mobile device, but other criteria can be also implemented.

As discussed in Section II-A, MobFogSim currently supports cold migration and (live) post-copy migration. However, the simulator can also be extended to support other migration techniques, e.g., pre-copy and hybrid. MobFogSim also introduces new Java classes to represent specific roles in a fog environment with mobile users. The *ApDevice* class extends the *FogDevice* class from iFogSim to implement the behaviour of access points, which manages wireless handoffs. The *MobileDevice* class also extends *FogDevice* to represent mobile devices. In iFogSim, both access points and mobile devices are implemented as instances of *FogDevice*. Moreover, MobFogSim introduces the *Coordinate* class for specifying the location of entities during simulation.

MobFogSim is also integrated with Simulation of Urban MObility (SUMO) [12]. This enables simulation to be carried out using data from realistic mobility databases. An example is the Luxembourg SUMO Traffic (LuST) database [13], which contains data from vehicle mobility in Luxembourg. The workflow is as follows. The LuST database is provided as XML input to SUMO. The latter interprets mobility data and produces a csv file for each simulated vehicle. The information reported in each csv file is: (i) position x and y on the map; (ii) speed in metres per second; (iii) direction in radiants; and (iv) simulation time. This new database is hence used by MobFogSim as a basis to define users' mobility. However, to adapt this database to its mobility model, our simulator makes some modifications, such as conversion of speed to kilometres per hour and conversion of direction to the eight main cardinal points.

B. Preliminary simulations

In this section, we highlight some of MobFogSim capabilities by evaluating the impact of user mobility and speed on service migration in fog computing. This extends previously reported results from [7]. This scenario motivates our proposal to introduce network slicing as a possible way to improve performance of service migration for mobile users in a fog environment.

1) Simulation setup: The simulation settings in this section are based on values extracted from a physical testbed that was used in [7] to validate our simulator. User mobility patterns are instead taken from LuST. In the testbed configuration, the user's device was an ASUS Zenbook UX331UN notebook, and the fog nodes were represented by two Raspberry Pi 3 Model B devices. Two network conditions were emulated in the testbed: (i) a *slower network* presenting, on average, a latency of $122.95\,\mathrm{ms}$ and a throughput of $11.34\,\mathrm{Mbps}$; (ii) a *faster network*, presenting a latency of $6.94\,\mathrm{ms}$ and a throughput of $72.41\,\mathrm{Mbps}$. To obtain values for the slower network, we considered a fixed device connected through Ethernet and a smartphone connected to the Internet through 4G/LTE. For the faster network, instead, we employed two fixed computers belonging to a bridged LAN and located $1\,\mathrm{km}$ far apart. Further details related to the testbed can be found in

[1] See https://github.com/diogomg/MobFogSim. Last accessed: August 24th, 2020.

TABLE I: Simulation parameters and their values based on physical testbed.

Parameter	Value
Client execution speed (*mips*)	2901 MIPS
Client disk usage (*size*)	4 MB
Server disk usage (*size*)	412 MB
Max. user device capacity	46534 MIPS
Max. fog node capacity (*mips*)	3234 MIPS
Number of instructions executed by client (*tupleCpuLength*)	966 million
Number of instructions executed by server (*tupleCpuLength*)	2439 million
One-way latency between fog node and user device (*UplinkLatency*)	4.78 ms
One-way latency between fog nodes in the slower network (*lat*)	61.48 ms
One-way latency between fog nodes in the faster network (*lat*)	3.47 ms
RAM requirement of client (*ram*)	49 MB
RAM requirement of server (*ram*)	128 MB
Server execution speed (*mips*)	281 MIPS
Throughput from user's device to fog node (*upBw*)	13640 kbps
Throughput from fog node to user's device (*downBw*)	13363 kbps
Throughput between fog nodes in the slower network (*bw*)	11612 kbps
Throughput between fog nodes in the faster network (*bw*)	74148 kbps

[9] and [7]. Table I summarises the settings of the simulated environment, which are based on the testbed, as well as the names of the MobFogSim variables in brackets. Computing resources are described in terms of MIPS (Million Instructions Per Second) – this metric is used to maintain compatibility with iFogSim.

Additional parameters used in the simulation include: 144 uniformly distributed fog nodes placed on a map of 10x10 km. Each fog node is connected to one access point that has a signal coverage of $500\,\mathrm{m}$. In our simulations we considered containers as virtual environment to encapsulate fog services, and we employed (live) post-copy technique to migrate such containers. The chosen migration strategy selects the fog node that presents the lowest latency to the user. User mobility patterns in the simulations were selected from realistic traces of urban buses in Luxembourg [13]. Each bus has a different route and speed, but buses move on average at a speed of $22.3\,\mathrm{kmph}$ and finish their routes in $26.44\,\mathrm{min}$. Aiming to improve mobility scenarios in terms of different speed values considered, we built two additional datasets. Both these datasets have the same routes as the original, except the speeds of buses, which were changed to two and three times the original speed. These three datasets allow us to perform a comparison within the same realistic routes but using different user speeds. The simulation scenarios described above allow the evaluation of the impact of different conditions, such as changes in user speeds and network resources, on the service migration process. Table II summarises the values used in the simulations.

2) Results: The impact of user mobility on resource management in fog computing can be measured through different metrics. In this work, we consider the number of migrations, the average migration time and the average delay to the container. The number of container migrations along a users' routes is significantly affected by the users' speed. As presented in Fig. 2, faster users tend to commit fewer migrations under both network conditions because such users reach their destinations sooner.

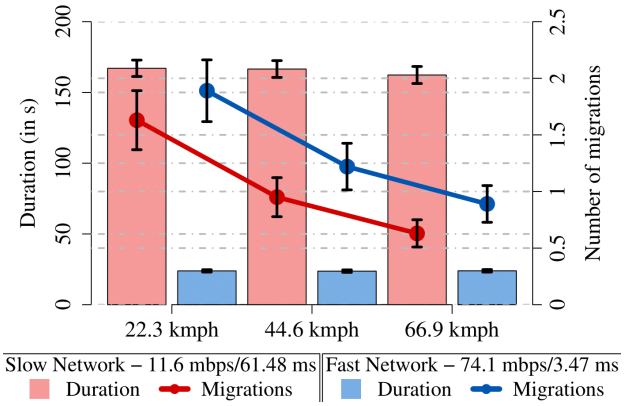


Fig. 2: Average number of migrations and migration time, based on [7].

Even though user speed decreases the number of migrations, it does not impact on migration time. Fig. 2 presents the results for this metric, with a confidence interval of 95%. However, the specific network condition has a considerable influence on migration time. As expected, the average migration time supported by the faster network is lower than that in the slower network. These results are, on average, 85% lower.

Another relevant metric in the context of fog computing is latency to the fog service. Fig. 3 shows the results for this metric. Simulations indicate that users that are two and three times faster than the original speed present a latency that is almost 30% higher if considering the slower network. Furthermore, the slower network provides, for any user speed, a latency that is more than 10 times higher than that under the faster network.

These results present the impact of user speed on container migration in fog computing, especially in terms of latency and number of migrations. In this context, faster users require special attention, especially if using network links with fewer resources. On one hand, migration times are not significantly affected by a users' speed, however the number of migrations decrease for faster users because there is not enough time to commit more container state during user movement.

Improving container migration for faster users may provide them with a better quality of experience while using the fog

TABLE II: Additional parameters and their values for simulation in MobFogSim.

Parameter	Value
Size of container execution state during live migration	12.8 MB
Access point coverage (radius)	500 meters
Number of fog nodes	144
Density of fog nodes per access points	1:1
Migration strategy	Lowest latency
Migration point policy	Static (40 meters)
Average user speeds	22.3, 44.6, and 66.9 Km/h
Number of users per simulation	1
Number of different user mobility patterns	100

infrastructure. Increasing the link bandwidth among fog nodes may not be a cheap solution. Providing these users with some priority in the use of the network may mitigate the problem. By introducing network slicing in this context, fog systems can allocate dedicated resources for these users on demand.

IV. Network Slicing in MobFogSim

In this section, we describe how we extended MobFogSim to model network slicing. Infrastructure resources in MobFogSim can be categorised in network and cloud/fog resources. The latter, which refer to storage and computing resources from cloud servers and fog nodes, are provided as VMs or containers. Network resources in MobFogSim are currently expressed in terms of the bandwidth available in the links. These links are from access and transport networks. The transport network consists of the interconnections among fog nodes, while the access network is related to the wireless interfaces that connect users to the fixed infrastructure.

We implemented network slicing in MobFogSim by orchestrating the distribution of resources among slices. All the slices thus share the same physical infrastructure in the simulator; however, each slice reserves a pool of resources. Resources reserved by a slice are dedicated to users within that slice. We achieved slicing support in MobFogSim by further extending our simulator with two elements. The first element is a new Java class called *Slice*, which is located in the *org.fog.placement* package. This class defines the portion of network bandwidth that is granted to a slice. Values are currently expressed as a percentage of the overall bandwidth. However, modifications can be made in the simulator to use absolute values. Moreover, *Slice* specifies the groups of users that have access to its slice resources. Multiple groups can be indeed assigned to the same slice. The second element introduced in MobFogSim is a new attribute, called *groupid*, associated with the *MobileDevice* class. This attribute is used as a tag to identify the group of users to which that specific mobile device belongs. These groups are just symbolical and represent the different users' characteristics and requirements defined in the simulation setup. Allocation of VMs/containers in this extended version of MobFogSim follows the same rules as in the previous release [7]. Each user is provided with a VM/container described in terms of processing and storage attributes. It is possible to grant different cloud/fog resources based on *groupids*. Thus, the *Slice* class in MobFogSim describes the list of fog nodes and their respective resources that one slice may access. However, an analysis of the impact of different amounts of cloud/fog resources for each slice is out of the scope of this work.

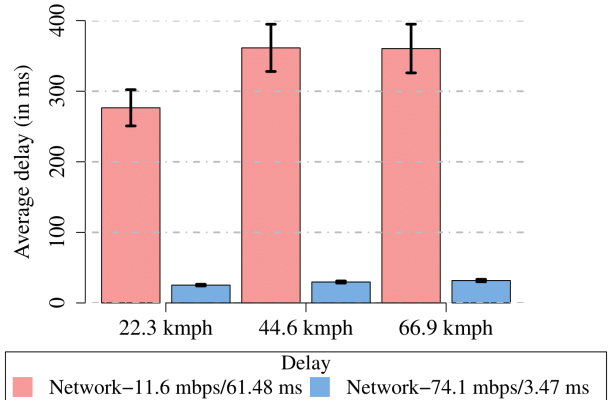


Fig. 3: Average delays, based on [7].

In MobFogSim, the overall network bandwidth is split among slices in terms of packets. Namely, based on the portion of bandwidth assigned to a slice (as specified by the *Slice* class), the simulator selects that percentage of traffic for that slice. For instance, if a slice instance claims 30% of the whole bandwidth, 30% of all packets transmitted on transport and access links are reserved to users of that slice.

Our extension of MobFogSim also supports dynamic network slicing. We highlight however that only bandwidth resources can be dynamically reallocated among slices in the current release of our simulator. Hence, idle bandwidth resources may be reassigned at runtime to other slices that need more bandwidth to attend their users' requirements. It is worth noting that resource reallocation is not immediate, as it always incurs a reallocation time due to computing overhead. We modelled this time as a parameter rather than a constant in our simulator, since each simulation setup may experience a different reallocation time. Specifically, the value of this parameter can be either inserted by MobFogSim users or calculated by a function in the simulator. As a result, static network slicing does not allow reallocation of resources; however, such resources are immediately available to slices once they request them. On the other hand, dynamic network slicing enables the reallocation of resources at runtime, but this reallocation experiences a delay.

V. Static Network Slicing for Mobile Users

In this section, we evaluate the impact of static network slicing on service migration to support mobile users within a fog computing environment. Despite the slicing support in MobFogSim covers network, storage, and computing resources, in this paper, we focus on the first aspect and leave the other two for future work. As a result, all mobile users received the same amount of computing and storage resources in the form of containers. We however considered different distributions of users and network resources (i.e., bandwidth) in our simulations. We highlight that, while this section focuses on static network slicing, Section VI instead considers the case of dynamic network slicing.

A. Simulation setup

Fig. 4 presents both the physical and logical architectures used in our experiments. The physical architecture of our fog computing environment is composed of two fog nodes connected by routers in the access and transport networks. Users are connected to the fog computing infrastructure through wireless access. Two network slices, namely Slice 1 and Slice 2, are created on top of the shared physical infrastructure.

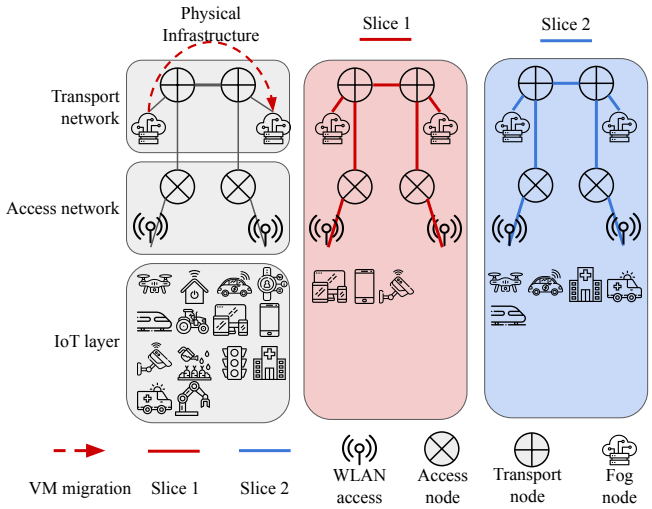


Fig. 4: Physical infrastructure and network slices used in our scenarios.

All the considered scenarios consisted of 20 users whose mobility caused one migration of container each from one fog node to the other. All 20 users started from the same point in the map and moved with the same direction and speed. As a result, all migrations started simultaneously. Each simulation finished when container migrations ended for all the users. The input values used in our simulations were the same considered in the preliminary experiments and are summarised in Table I and Table II. Despite these points: the simulation scenario is only evaluated on the faster network; the number of fog nodes is restricted to 2; the number of users is increased to 20; and all users present the same constant direction and speed mobility pattern.

In our experiments, we distributed the 20 users in three different ways between the two network slices. Specifically, in Slice 1, we deployed the first three powers of two (starting from two), thus producing the following three distributions of users between Slice 1 and Slice 2, respectively: (i) 2 against 18; (ii) 4 against 16; and (iii) 8 against 12. We also considered three different distributions of network resources between slices, by first assigning 50% of the resources to each slice and then adding/removing 20% of the resources twice. Therefore, we built the following three distributions of network resources between Slice 1 and Slice 2, respectively: (i) 10% against 90%; (ii) 30% against 70%; and (iii) 50% each. By combining the distributions of users and resources, we thus considered nine different scenarios in our simulations. We also highlight that the transmission of container's data was performed using packets up to 1 MB.

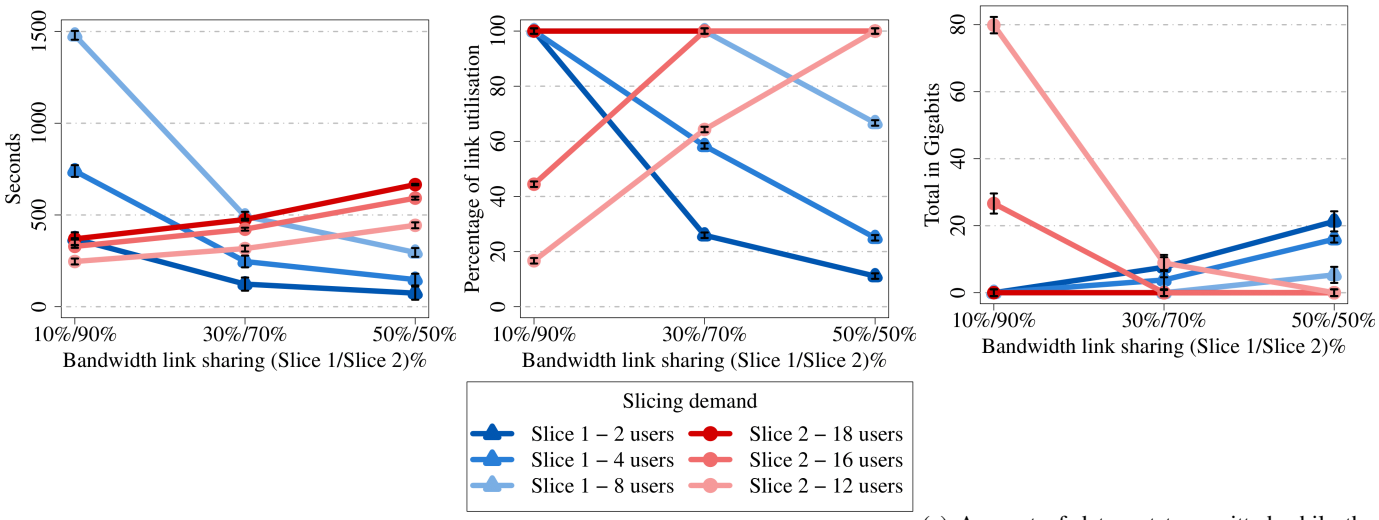
For each scenario, we considered the following three metrics: (1) total migration time; (2) percentage of link utilisation throughout the simulation; and (3) the amount of data that could be transmitted while the link is idle. Link utilisation (2) is calculated based on the average amount of data sent through that link and is expressed in terms of percentage of the link capacity. The link used in this measure is the one that connects the two fog nodes and is used to transmit the containers' data during migration. Finally, the potential amount of data not transmitted (3) is calculated based on the time that bandwidth resources are allocated but not used. We performed container migrations according to the post-copy technique and run 25 simulations for each scenario. Results are shown with a confidence interval of 95%.

B. Results

Considering that each container in our simulations has a data size of 128 MB, 20 users simultaneously requesting migrations lead to 2.56 GB of data transmitted in parallel. Using post-copy migration in a traditional network, container migration would take 369 s on average. Through network slicing, it is possible to discriminate the traffic for different users and introduce some priority for them. Based on our simulation scenarios, we analyse the impact of the balance between users' requirements and resource allocation on the migration process.

Fig. 5a presents the average total migration time. The blue lines represent the performance of Slice 1, while the red lines are related to Slice 2. It is possible to notice how crucial is the portion of bandwidth dedicated to each slice. In the first scenario where only 10% of the bandwidth was allocated to Slice 1, the average migration time for 4 users in Slice 1 (739 s) is almost twice that of 16 users in Slice 2 (328 s). The performance of Slice 1 with 8 users is even worse: the average migration time in that slice (1479 s) is almost six times that of the remaining 12 users in Slice 2 (246 s). It is clear that this 10%/90% scenario presents an unfair distribution of network resources. However, average migration time tends to be more propitious to Slice 1 as the bandwidth is more equally shared. In the 30%/70% scenario, migration times for Slice 1 and Slice 2 are more similar. In the last scenario (50%/50%), migration times for Slice 2 are between 130% and 900% longer than those for Slice 1 because fewer users belong to the latter slice.

Fig. 5b presents the results of link utilisation. Firstly, it is possible to note how links that show worse migration times in each scenario (see Fig. 5a) are used all the time throughout the simulations. Those are indeed the links where the last container terminates its migration, which triggers the end of the simulation. Allocation of network resources is efficient when there is a high link utilisation (i.e., no wastage of resources) and migration times meet users' requirements. However, the allocation may be inefficient in two other cases. The first case is one in which links are at their limit of utilisation, but migration times are far above the required ones. An example of this case is that of 8 users assigned to Slice 1 with 10% of bandwidth allocated. The second inefficient case is that in which network resources are idle (i.e., not utilised after all migrations for that slice terminated) for a long period. Those resources, which could be leveraged to boost migrations for other slices, are indeed wasted. For instance, when 12 users are assigned to Slice 2 with 90% of network bandwidth, only 20% of those resources are actually used. Fig. 5c can help to better understand this concept. It presents



(a) Average migration time.

(b) Percentage of link usage.

(c) Amount of data not transmitted while the link was idle.

Fig. 5: Performance of static allocation of network slices for mobile users.

the amount of data that could be potentially transmitted while network links are idle. As already discussed, the scenarios where resources are not fairly balanced present the higher index of wasted bandwidth. This unfair balance is justified by the objective to decrease the migration times for high-priority users. However, static slicing is not able to reallocate idle resources to other slices that need them. For instance, the scenario in which 12 users are assigned to Slice 2 with 90% of bandwidth achieves average migration times of 246 s. However, once migrations for Slice 2 are finished, its network resources are not reassigned to Slice 1. This leads to average migration times for Slice 1 to be 1479 s, while about 80 Gb of data could have been potentially transmitted for that slice.

The results in this section provide two insights. Firstly, network slicing can help to prioritise users and reduce times to migrate their containers, thus allowing them to perform more migrations and keep their services closer to them. Secondly, static network slicing causes an inefficient allocation of resources, which cannot be reassigned to other slices to boost their migrations. In the next section, we evaluate how dynamic network slicing solves this problem.

VI. Dynamic Network Slicing for Mobile Users

This section analyses the impact of dynamic network slicing on container migration for mobile users in the fog. With dynamic network slicing, resources can be reassigned at runtime to other slices that demand them.

A. Simulation setup

The simulation setup defined for this set of experiments is the same as that reported in Section V. The only difference is related to how idle resources are dealt with. In the previous section, the portion of bandwidth that is allocated to a slice is guaranteed to that slice for the entire simulation. In this section, instead, it is possible to reallocate resources on demand, following the concept of dynamic network slicing. In this work, we only reallocate idle resources. Therefore, once all the migrations for a slice finish and the link becomes idle, that part of the bandwidth is completely reallocated to the other slice.

As described in Section IV, the process to reallocate resources among slices has a computational cost that results in time overhead. That cost, which is set as a parameter in MobFogSim, depends on the characteristics of the physical and logical networks. For these experiments, we defined this cost as 1 s. However, different values may impact overall performance.

The experiments in this section are organised in two parts. In the first one, we defined users in Slice 1 as higher-priority users. Based on that, once bandwidth becomes available in Slice 2, it is reallocated to Slice 1. The opposite does not occur. In the second part of the experiments, Slice 1 has no more higher priority. Then, both slices can benefit from reallocation of resources, once these become idle. Results are relative to 25 runs per scenario and shown with a confidence interval of 95%. We used post-copy technique to migrate containers.

B. Results

Fig. 6a presents average migration times in slicing scenarios with unidirectional reallocations. In these experiments, Slice 1 receives the available bandwidth from Slice 2 once this resource becomes idle. This procedure helps Slice 1 in those scenarios in which, with static slicing, it presented worse performance than Slice 2 (see Fig. 5a). Scenarios where Slice 1 receives 10% of the resources and serves 4 and 8 users, and the scenario in which it receives 30% of the bandwidth and serves 8 users present average migration times that are around 50%, 25%, and 75% of those obtained with static allocation,

respectively. Using dynamic slicing, the worse result for Slice 1 is limited to 369 s compared to 1479 s with static slicing. Withdrawing idle resources from Slice 2 results neither in an improvement nor in a worsening for that slice. Considering all scenarios, dynamic slicing leads to an average migration time of 406 s, which is almost 15% less than the 469 s of static slicing.

There are also some scenarios in which Slice 1 presents idle resources. Those resources could be used to improve performance for users in Slice 2. In this second group of experiments, also Slice 1 thus shares its idle resources with Slice 2. Fig. 6b shows average migration times in slicing scenarios with bidirectional reallocations. Under these conditions, Slice 1 presents the same performance as that in Fig. 6a. However, also Slice 2 now receives a boost in its worst cases. Both Slices are now limited to 370 s in their worst cases. The average migration time for Slice 2 presents an improvement, being almost 15% less than that with the static allocation of resources. With bidirectional reallocation, link utilisation reaches nearly 100% for both slices. Therefore, the dynamic network slicing achieves an efficient resource allocation, as it avoids underutilisation of network links while reducing migration times for mobile users.

VII. Open Challenges

This work described our extension of network slicing support in MobFogSim. Besides, it reported an experimental evaluation of static and dynamic network slicing as ways to improve container migration performance for mobile users in fog computing environments. Still, some open issues need further investigation, such as:

- *Slicing reallocation overhead* - once a slice is created, and an amount of resources is allocated to it, there is a computational cost to resize it. This cost depends on the physical and logical network characteristics. Network appliance from different brands and different software to manage it present different efficiency rates to reallocate network slices. The cost of this reallocation process may impact on the reallocation performance. Once a slice asks for more resources, such resources should be reassigned as soon as possible. If that is a slow process, predictive demand can be studied.
- *Slicing reallocation frequency* - the frequency at which reallocation should occur is another open challenge. This frequency is strictly related to the reallocation cost: the lower the cost is, the higher the reallocation frequency can be. Moreover, frequency could be adapted to the priority of slices. For instance, high-priority slices could lend their resources with a reallocation frequency that is lower than that of low-priority ones. In this way, high-priority slices would incur a lower reallocation cost when they receive their resources back. Further analysis of slicing reallocation frequency is needed.
- *Impact of dynamic slicing over time* - this work presented an experimental analysis of dynamic slicing in a small-scale context with two fog nodes. In a more complex and realistic scenario, users with different mobility patterns in a wider fog infrastructure may create dynamic demand over different areas and periods of time. Under such conditions, slices may need to deal with different requirements over time, and the impact of resource sharing and slice resizing needs to be further studied.
- *Dynamic slicing on not-only idle resources* - reallocation of only idle resources may not satisfy the demands of some slices. In a more comprehensive scenario, busy resources should be reallocated as it happens for idle ones. To this purpose, the architecture should rank slices, take resources that are currently in use by lower-priority slices, and give such resources to higher-priority ones that need them. However, this procedure is not trivial and should be further analysed.
- *Evaluation of end-to-end slices* - this work evaluated network slicing in terms of bandwidth resources to enhance container migration performance. However, MobFogSim also models slicing of computing and storage resources. In such a dynamic scenario present by mobile users, an evaluation of end-to-end dynamic slices, which includes resources and services from access, transport, and core network, are needed.

VIII. Related Work

This section overviews the related work in a twofold way. It first presents the available simulators for fog computing and discusses the novelties of MobFogSim, among which stands the implementation of dynamic network slicing. Then, Section VIII-B reports the works that propose network-slicing-based approaches to improve the use of fog computing resources for supporting mobile users.

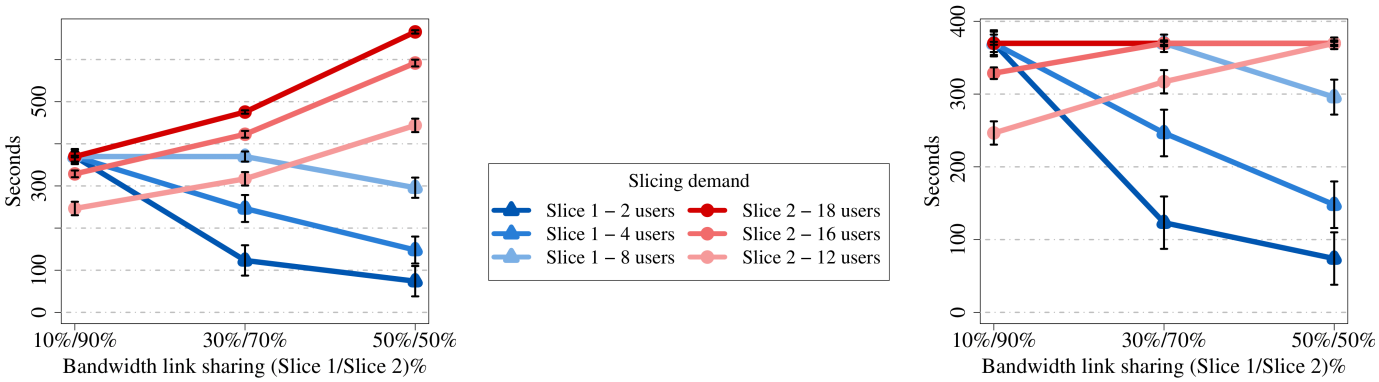
A. Fog computing simulators

A number of simulators is available for fog computing. Table III summarises the main contents of this section.

Yet Another Fog Simulator (*YAFS*) [14] is a Python tool that is particularly tailored to model network failures and hence allows to design robust networks or evaluate service placement strategies in case of failures. YAFS models network failures in two possible ways: (i) by dynamically creating/deleting fog nodes and network links; (ii) through *custom processes*, i.e., functions that are invoked at runtime for the implementation of real events. YAFS models device mobility but does not model VM/container migration or dynamic network slicing.

VirtFogSim [15] does not model VM/container migration nor dynamic network slicing. However, it dynamically tracks the application energy-delay performance against abrupt changes due to failures or device mobility, e.g., mobility-induced changes to the available uplink/downlink bandwidth. The most interesting feature of VirtFogSim is the modelling of cellular network access, which is useful when simulating 4G/5G scenarios. At the moment of writing, VirtFogSim is the only fog computing simulator that does this.

FogNetSim++ [16] focuses on network modelling. The other simulators do not (or only partially) take into con-



(a) Average migration times using dynamic slicing to prioritise one slice.

(b) Average migration times using dynamic slicing to reallocate resources for both slices.

Fig. 6: Performance of Dynamic Network Slicing

TABLE III: Comparison among the fog computing simulators.

Simulator	Mobility/Handoff	VM/container Migration	Dynamic Network Slicing	Programming Language
YAFS	•	-	-	Python
VirtFogSim	•	-	-	MATLAB
FogNetSim++	•	-	-	C++
EdgeCloudSim	•	-	-	Java
MobFogSim	•	•	•	Java

sideration real-network properties. They simulate an ideal network with no packet loss, congestion or channel collision. Instead, FogNetSim++ extends OMNeT++[2] to model these aspects. Moreover, it includes popular communication protocols for simulation, such as TCP, UDP, MQTT, and CoAP. FogNetSim++ models mobility; however, it does not model VM/container migration and dynamic network slicing.

EdgeCloudSim [17] is a simulator for fog computing environments that, like iFogSim, extends CloudSim. Network delays are modelled in EdgeCloudSim more realistically than in iFogSim, where delays are always fixed. EdgeCloudSim, instead, updates network delays based on the current network load. However, EdgeCloudSim implements less functionalities than iFogSim. For example, energy consumption, operational costs, and pricing are all missing. Device mobility is modelled; however, VM/container migration and dynamic network slicing are not.

None of the discussed simulators model VM/container migration, which however is important for supporting device mobility but also for other reasons such as load balancing. We implemented *MobFogSim* [7] to fill this gap in literature. MobFogSim is currently the only simulator that models VM/container migration in fog computing networks. Furthermore, this work extends MobFogSim with the implementation of dynamic network slicing, which is not provided by any other fog computing simulator. MobFogSim thus inherits all the functionalities from iFogSim, extending this simulator with mobility modelling, VM/container migration, and dynamic network slicing.

[2]See https://omnetpp.org/. Last accessed: August 26th, 2020.

B. Network slicing in the fog to support mobile users

Some literature focuses on the use of network slicing to support mobile users in fog computing contexts. Presenting a better resource allocation scheme for mobile users, the work in [18] proposes a logical architecture for network-slicing-based 5G systems. Considering mobile scenarios with different network requirements, the authors focus on the radio allocation to improve handover mechanisms. Even though wireless handover contributes to service continuity, network slicing can potentially also improve a user's data migration between fog nodes, which is not discussed in [18].

Authors in [19] present a comparison between static and dynamic slice allocation, proposing two solutions for resource allocation, based on integer linear programming and heuristic approaches, aiming to maximise the number of attended requests. The slice requirements take into consideration network and computational resources, but only link capacity requirements change over time. That variability of user requirements makes the experiment closer to real cases. However, in [19] user requests are randomly generated, which may not represent real mobility scenarios.

The work in [20] proposes a network slice scheduling scheme for vehicular fog networks. The proposed scheme aims at handling the unpredictability of network traffic and fog nodes availability, which is caused by the high mobility of vehicles. It is worth mentioning that the fog environment considered consists of fog nodes that can be devices located at the network edge (e.g., cellular base stations) as well as end user vehicles, in accordance with the Vehicular Fog Computing (VFC) paradigm [21].

Authors in [22] focus on the containerised services that are assigned to a group of users and run in the fog within a network slice. Such services, which represent the computation component of a network slice, need to be migrated across the fog infrastructure to follow their users. The migration of services belonging to a network slice is defined by authors as *slice mobility*. A testbed consisting of two fog nodes is set up to perform a preliminary performance evaluation of container migration. We highlight that network slicing and container migration are both considered. However, network slices are not examined as a way to improve container migration performance, but they are instead the object of migration.

IX. Conclusions

Fog computing systems need to accommodate users with very different mobility patterns and application requirements. Network slicing could play an important role in dealing with such a diverse and heterogeneous environment. Network slicing creates several logical networks on top of the physical one, where each logical network is tailored to address the requirements of a specific group of users and applications. This paper presents the role of static and dynamic network slicing to deal with different mobile user priorities in fog computing environments. Furthermore, it describes the extensions made in MobFogSim to support such a scenario. Results show that both static and dynamic slicing can introduce traffic priority to deal with different service migration requirements. Network resources may present underutilisation in static slicing in some periods. On the other hand, the capacity of dynamic slicing to scale up or scale down provides a better utilisation of network resources. However, dynamic resource reallocation is sensitive to a computing overhead, and resource requests may need to wait for reassignments to occur. Furthermore, MobFogSim is presented as a potential tool to simulate scenarios using network slicing for fog computing environments. As future works, we plan to address the open challenges listed in Section VII, such as the problem of slicing reallocation frequency and that of dynamic slicing on not-only idle resources. Besides, we will introduce dynamic reallocation of cloud resources in MobFogSim.

Acknowledgments

This work is part of the INCT of the Future Internet for Smart Cities (CNPq 465446/2014-0, CAPES 88887.136422/2017-00, and FAPESP 2014/50937-1). The authors thank CNPq grant 420907/2016-5 and FAPESP #2015/24494-8. Diogo Gonçalves is partially funded by Coordenação de Aperfeioamento de Pessoal de Nível Superior Brasil (CAPES) Finance Code 001. The authors also thank the Italian Ministry of Education and Research (MIUR) in the framework of the Crosslab project (Departments of Excellence).

References

[1] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog Computing for the Internet of Things: A Survey," *ACM Transactions on Internet Technology*, vol. 19, no. 2, Apr. 2019.

[2] C. Puliafito, E. Mingozzi, and G. Anastasi, "Fog Computing for the Internet of Mobile Things: Issues and Challenges," in *IEEE 3rd International Conference on Smart Computing (SMARTCOMP)*, May 2017, pp. 1–6.

[3] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards Virtual Machine Migration in Fog Computing," in *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Nov. 2015, pp. 1–8.

[4] K. Velasquez, D. Perez Abreu, D. Goncalves, L. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, "Service Orchestration in Fog Environments," in *IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug. 2017, pp. 329–336.

[5] Z. Zaidi, V. Friderikos, Z. Yousaf, S. Fletcher, M. Dohler, and H. Aghvami, "Will SDN Be Part of 5G?" *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3220–3258, Fourthquarter 2018.

[6] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, thirdquarter 2018.

[7] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "Mobfogsim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, p. 102062, 2020.

[8] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT Edge Computing with Lightweight Virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, 2018.

[9] C. Puliafito, C. Vallati, E. Mingozzi, G. Merlino, F. Longo, and A. Puliafito, "Container Migration in the Fog: A Performance Evaluation," *MDPI Sensors*, vol. 19, no. 7, Mar. 2019.

[10] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "MyiFogSim: A Simulator for Virtual Machine Migration in Fog Computing," in *ACM 6th International Workshop on Clouds and (eScience) Applications Management (CloudAM). Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, 2017, pp. 47–52.

[11] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, May 2017.

[12] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban Mobility: An Overview," in *3rd International Conference on Advances in System Simulation (SIMUL)*, 2011.

[13] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research," in *IEEE Conference on Vehicular Networking (VNC)*, Dec. 2015, pp. 1–8.

[14] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A Simulator for IoT Scenarios in Fog Computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[15] M. Scarpiniti, E. Baccarelli, and A. Momenzadeh, "VirtFogSim: A Parallel Toolbox for Dynamic Energy-Delay Performance Testing and Optimization of 5G Mobile-Fog-Cloud Virtualized Platforms," *MDPI Applied Sciences*, vol. 9, no. 6, March 2019.

[16] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "FogNetSim++: A Toolkit for Modelling and Simulation of Distributed Fog Environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, Oct. 2018.

[17] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An Environment for Performance Evaluation of Edge Computing Systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, Aug. 2018.

[18] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. Leung, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, Aug. 2017.

[19] M. R. Raza, M. Fiorani, A. Rostami, P. Öhlen, L. Wosinska, and P. Monti, "Dynamic Slicing Approach for Multi-tenant 5G Transport Networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 1, pp. A77–A90, Jan. 2018.

[20] K. Xiong, S. Leng, J. Hu, X. Chen, and K. Yang, "Smart Network Slicing for Vehicular Fog-RANs," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3075–3085, 2019.

[21] Z. Ning, J. Huang, and X. Wang, "Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[22] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network Slice Mobility in Next Generation Mobile Systems: Challenges and Potential Solutions," *IEEE Network*, vol. 34, no. 1, pp. 84–93, 2020.