Network Slice Mobility in Next Generation Mobile Systems: Challenges and Potential Solutions

Rami A. Addad[1], Tarik Taleb[1,3,4], Hannu Flinck[2], Miloud Bagaa[1], and Diego Dutra[5]
[1] Aalto University, Espoo, Finland
[2] Nokia Bell Labs, Espoo, Finland
'3 Oulu University, Finland
[4] Sejong University, Seoul, Korea
[5] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

***Abstract*—Network slicing offers numerous benefits, particularly the ability to deliver highly customizable services to new industry sectors that have been unserved or inadequately served by current mobile network operators. Among new industry use cases that are targeted by the fifth generation (5G) mobile systems, there exist scenarios that go beyond what the current device-centric mobility approaches can support. The mobility of low latency communication services, shared by a group of moving devices, e.g., autonomous vehicles that share sensor data, is a prime example of these cases. These use cases' demands for ultra-low latency can be addressed by leveraging the Multi-Access Edge Computing (MEC) concept, techniques for live migration of virtual resources, Software Defined Networking (SDN), and network slicing. In this paper, we define different slice mobility patterns, different methods for grouping users, and different triggers for network slice mobility. Furthermore, we evaluate the mobility of services and network slices based on the simultaneous migrations of multiple containers.**

***Index Terms*—5G, Network Slice, Network Softwarisation, Migration, NFV, SDN, and MEC**

I. Introduction

The 5^{th} generation mobile networks ($5G$) will go beyond providing only high data rates for mobile users, as it will be a platform for a wider communication ecosystem for the Internet of Things and machine-type communications applications [1]. The services of new vertical industries, e.g., automotive, e-health, public safety, and smart grids impose unique requirements that will push the envelope for high performance, scalability, and availability. To achieve such ambitious goals, $5G$ has been re-architected from the ground up in comparison to the previous mobile network generations. Decoupling logical network functions from the physical infrastructure forms the basis of deployment of self-contained, programmable and customizable networks [2].

The Network Function Virtualization (NFV) and Software Defined Networking (SDN) are the fundamental technologies to implement the separation of logical network functions from the infrastructure leading to network softwarization [3]. SDN offers programmability of the connectivity between the network functions while NFV provides means to define, instantiate and manage virtualized network functions that are required to create multiple logical (virtual) networks. The sharing of the same underlying infrastructure among isolated and self-contained networks leads to the key concept of network slicing [4].

Network slicing is an active research field within the academic community and among the different standards development organizations (SDOs), such as the Next Generation Mobile Network Alliance (NGMN), Third Generation Partnership Project (3GPP), and International Telecommunication Union – Telecommunication Standardization Sector (ITU-T). In the scope of this paper, we define a network slice as an end-to-end (E2E) logical network running on top of common underlying (physical or virtual) network, fully isolated, with independent control and management, and flexibly programmable on-demand to meet service level agreements (SLA) of a specific service. A network slice consists of computing and storage resources, associated with virtual networks, possibly composed of multiple virtual sub-network segments; and may span across multiple technological as well as administrative domains. Mobile users can connect to multiple slices simultaneously depending on the type of service they are using [5]. Furthermore, for latency-sensitive services, MEC [6] provides powerful service delivery with minimal delay. In a MEC environment, MEC servers, as well as network access points, can be sliced to serve multiple slices with very different service characteristics.

However, as users move with their mobile devices from one domain to another, their ongoing mobile communication and service sessions running on at least a network slice may suffer a drop in QoS if not a total disconnect. This may happen if their ongoing network service has to be served by instantiating a totally new network slice of the same type at the destination network (new radio resources, backhaul, computing, and storage). This leads to a network slice that has to support moving its computation around following its users' mobility pattern, additionally, the slice itself may need to adjust its resource allocation, adding more resources or freeing unused ones, these requirements lead us to the definition of the concept of slice mobility.

Each instantiated network slice should ensure service continuity for its end-users. Intuitively, a simple service replication would solve the issue, yet not all services are stateless (i.e., service that do not save user session/context). It is worth noting that the migration of stateless services is less demanding

than the migration of stateful (i.e., service that saves user session/context) services [7]. To support migration of stateful services, separate mobility mechanisms must be designed and implemented to ensure consistency of the service context. In addition, the resources of a slice could become saturated and over-consumed due to a sudden increase in the service demand, thus actions for slice scaling or breathing should be supported. Fortunately, the same actions that are needed for *slice mobility* suit well for slice scaling. Based on these observations, the contributions of this paper are:

- The introduction of a number of slice mobility patterns to optimally manage and use slices with their allocated resources, while leveraging the ideas of Follow me Edge concept introduced earlier in [8]–[10].
- The examination of service migration capabilities needed for slice mobility inside one Infrastructure as a Service (IaaS) cloud domain, that we then extend to cover slice mobility across multiple IaaS domains.
- The definition of several key slice mobility triggers and User Equipment (UE) grouping methods to enable efficient slice mobility needed for the described slice mobility use cases.
- A preliminary evaluation of two key enabling technologies for slice mobility patterns: system virtualization with containers and SDN to allow a fast and seamless allocation of resources to a network slice.

The remainder of this paper is structured as follows. Section II summarizes the fundamentally related research work. Section III gives an overview of different slice mobility use cases. Section IV introduces the key slice mobility triggers and the different methods for grouping users besides slice mobility patterns. Section V focuses more on the enabling technologies for the proposed solution and the primary evaluation setup for the proof of concept. Finally, the paper concludes in Section VI.

II. Related Work

The authors of [11] focus on extensions for NFV orchestration that provide tailored support for mobility and QoS/QoE for network slices whilst ensuring efficient utilization of the substrate network resources. They present several ways to classify and select mobility management (MM) schemes based on the context of the service or type of a network slice. Terminal speed, session continuation requirements, and stability of the endpoint addresses are the main criteria for selecting a mobility scheme. The authors propose slice specific MM schemes through the creation of context-dependent configurations of the instantiated network functions. Our envisioned approach differs from this one by focusing on moving slices along with their services based on mobility triggers and grouping of respective UEs.

As previously discussed the migration process is one of the key enablers of the proposed slice mobility concept, where a running virtual instance (i.e., Virtual Machines (VMs) or containers) is migrated across different host machines (i.e., within the same IaaS platform or across multiple IaaS) without disrupting the service. Clark et al. [12] introduce the writable working set concept and use it to design a pre-copy based migration procedure that enables the live migration of VMs for the Xen Virtual Machine Manager (VMM). Mann et al. [13] present a network architecture that provides a layer-agnostic and seamless live and off-line VM mobility across multiple data centers. They leverage SDN and use the principle of location independence in order to handle the inter-data center limits. They obtain better results compared to the default layer 2 networks; in some tests, their solution outperforms the default approach by up to 30%. Tay et al. [14] evaluate the migration performance of Containers and VMs showing that the use of system containers can achieve the same, or even better, capabilities than a VM without its high overhead.

With respect to first cited work, in this study, we introduce the slice mobility patterns, and jointly emphasize the key enabler triggers and the grouping methods of the end users. While the remaining listed related works are used as a basis for enabling the desired slice mobility. Seeing that new use-cases entrance will beget a highly mobile environment, increase the core network traffic and reduce the latency, this work is a must for achieving the 1 ms latency dream for the upcoming 5G mobile systems and beyond.

III. Slice mobility use cases

Network slicing enables customization of network services and resources for the needs of different use cases. The same underlying infrastructure is shared among the different slices that add their own customized network functions and services to implement a dedicated service network. The services in a slice will have their own dedicated share of e.g. RAN resources, edge computing resources, and control plane resources that will be concatenated to form an end-to-end slice. Such a dedicated slice can be customized for a number of use cases. For example, to support autonomously moving equipment with mobility patterns and latency requirements that differ from a regular mobile phone, e.g. drones, robotic vehicles, or a fast-moving train carrying mobile users enjoying infotainment services. To ensure that the user mobility patterns match with the coverage and availability of the resources allocated to the slice, we introduce the concept of slice mobility that is implemented by carefully coordinated and organized live migrations of Virtual Network Functions (VNFs). In this section, we discuss use cases that clearly show the need for the proposed slice mobility paradigm that 5G networks should adopt. The first two use cases form a new mobility pattern not perceived before in the previous generation, they also express the perfect need for a mobile slice to follow the users (cars, and UAVs), ensure specific resource availability and continuous delivery during the mobility time. While the last use case is proposed to illustrate another aspect of the slice mobility patterns related to the partial mobility which will be detailed in the next section.

A. Drone Traffic Control

Unmanned aerial vehicles (UAVs) are the perfect example to illustrate devices with predictable paths and high mobility, showcasing the slice mobility use case. As an initial scenario, let us assume that Jonathan is a young researcher working on UAVs, and his mission is to perform a self-swarming control test over two university campuses. Jonathan has ordered a customized slice for his drone experiment. In addition to offering the connectivity to the drones, the slice is hosting virtual flight controllers for all the involved drones. The virtual flight controller is a software application that controls a corresponding drone and receives location information from the drone and radio usage information from the network. Virtual flight controller application is instantiated in a container. Each drone has its own instance of the controller and the controllers need to cooperate to group the drones into swarms in an orderly manner so they move as a group to the same direction. In his experiments, Jonathan noticed two possibilities: the first one is that the swarms of drones can have a synchronized mobility pattern where the swarm moves as a tight group outside the current service area or a second one where the drones move in smaller groups or one by one out from the initial service area depending on how tight the swarm is. In both situations, the containerized flight controller must follow its drone in order to provide the required low latency and accurate control of the drones. This use case leads to the notions of migrating the slice as a whole (i.e. full slice mobility) or the slice gets split into two service areas.

B. Autonomous Vehicles Support

Autonomous vehicles are thought to be one of the key verticals that would benefit from the upcoming 5G systems. Autonomous vehicles are expected to be served by mobile operators or car manufacturer-operated network slices. Safety of the passengers and the other public on the road mandates highly-reliable and low-latency connectivity to be an integral part of a slice for autonomous vehicles. A missed signal from the virtual data and processing analytics of a connected car (vDPACC) can cause an imminent danger situation that could result in physical damage or even a death. In addition to the support of low latency and high reliability, the slice must also have a redundancy that can be implemented at the VNF- and connection-level inside a slice or by a backup slice. The latter approach would lead to simpler overall service orchestration and configuration. When such a slice is relocated, the backup slice also needs to be modified or even moved as well. The slice mobility management mechanism must take into account the availability of backup connections, as well as the redundancy of the VNFs of the slice when migrating the slice and its services. The backup slice should not be relocated at the same time as the primary slice to ensure availability of the service.

C. Rapidly Changing Video Streaming Need

In this use case, a city hall is taking place in a form of a webinar. All the residents of the city have been invited due to the importance of the decisions to be made. The community has arranged its own slice for this mass meeting. The webinar is also attended by an audience on a train departing from the railway station of the city. The train leaves according to its schedule and enters soon to a neighboring area, or a tunnel, that has much less capacity to offer. The city hall videos streaming slice needs to be complemented with a new and temporal slice in the new service area that the train is passing by. This new slice solves the bottleneck by accommodating a special Content Delivery Network (CDN) capacity with a video streamer that aggregates multiple separate unicast video streams into a limited number of shared streams that are distributed to the traveling audience of the webinar. Once the train enters into an area with better infrastructure, the purpose-built video CDN slice can be released and the original slice can continue to serve the audience. This use case shows the need for slice splitting and slice merging leading to the concept of slice breathing discussed in the following sections.

IV. Key Enabler Triggers, Grouping Attributes, and Slice Mobility Patterns

As was discussed in the use case section slice mobility comes in a number of different variants or mobility patterns depending on how the slice is to be modified due to the changes in its service consumption, thus categorizing slice mobility patterns is a challenging issue. Based on what was introduced earlier, we identify following slice mobility patterns: full slice mobility, slice splitting, slice merging, slice shrinking and slice breathing. Slice mobility actions are based on a number of triggers characterizing the slice dynamics and its service consumption. We start by introducing the slice mobility triggers that initiate slice mobility actions. The mobility triggers, as well as slice mobility actions, operate on a set of different groupings depending on a service or its resources. We elaborate the key grouping attributes offered by the 5G network specifications that provided the bases for slice mobility patterns and mobility triggers.

A. Slice Mobility Triggers

In this step, we will identify, discuss and present several key triggers to enable different slice mobility patterns. Triggers broadly relate to the users' mobility, the availability of physical resources and network resources at the hosting edge cloud or federated cloud, resource efficiency utilization, service reliability, and security.

1) Group mobility trigger: This trigger can be considered as the main catalyst for slice mobility in a real-life environment. The group mobility trigger could be applied to all use cases cited-above (i.e., drone, autonomous cars, and video streaming). In this trigger, a group of users simultaneously move from a location to another one. For example, passengers on board a metro or train move simultaneously from a location to another (as in the use case of video streaming in a highly moving entity), which requires full slice mobility. The signal strength can be measured by the users and reported back to the access points. These measurements can be correlated and acted

upon already on the access points or deeper in the network, such as at the mobility management entity in the evolved packet system, access management function in the 5G core, or at the life cycle manager controlling the slice. The entity in charge of correlating these measurement reports will pull the group mobility trigger that will, in turn, launch the process of slice mobility to follow the mobility of that particular group of users, using the same slice and its services that were reported by the measurements.

2) Resource availability trigger: Edge clouds tend to always have fewer resources, i.e. network, processing, and storage capacity, than the centralized cloud. Due to the limited resources at the edge, the system level resource consumption must be monitored carefully. Once the upper limit of allowed total used capacity is reached, a "limited resources availability" trigger will be generated with a parameter indicating that the highest allowed resource consumption level of a certain type of system resources is reached. This trigger is sent to the slice life-cycle management to initiate migration of services from the highly-loaded edge cloud(s) towards the centralized cloud while keeping at the edge only delay-sensitive services. Alternatively, the services of the highly-loaded edge cloud could be migrated towards neighboring edge clouds if they have the necessary system-level resources available. Once the resource consumption level decreases below a given parametrized threshold, the edge cloud should send a "resource availability" trigger to indicate that there is room to accommodate more users and services for that particular type of resources.

Let's consider that the users are static and the network resources are exhausted by the requested data from a given group of users. In this situation, a scale-out operation needs to be considered. However, the slice is limited by the available physical resources of the edge cloud and the number of different slice migration strategies that need to be considered with potential impacts on the service QoE. One possible trade-off is to migrate some users or services to other edges or to the centralized cloud that is further away even if that may reduce their QoE.

3) Reliability trigger: The reliability trigger would be typically generated by the operation and maintenance protocols and supporting operations, administration and maintenance (OAM) systems of the access point or the edge cloud that monitors the healthiness of the connectivity. In case of a major disaster, there would be an abrupt interruption of connectivity towards the users or neighboring edge nodes or even to the centralized cloud. The services should be therefore simultaneously evacuated from one location to another as long as there is still some capacity left for that. All unnecessary actions should be deferred until the system snapshot has been replicated. This trigger will cause full slice mobility where all slices served by this access point or edge cloud node need to be migrated elsewhere to ensure the integrity of all valuable data.

4) Security trigger: Security is of vital importance for the networks since a compromised entity or service may result in a considerable damage to the whole infrastructure. For instance, in case a denial of service (DoS) attack occurs in a specific location "A", the services should be then shifted from that location to a more secure one. An intrusion detection system (IDS) could send a security trigger to start a slice mobility for the compromised slice. When the IDS detects an anomaly in a slice, this trigger is sent to the orchestrator that initiates the migration process for that slice.

5) Request overload trigger: Service request trigger is based on the number of simultaneous service requests stemming from groups of users requesting the services available in an existing slice. As a potential number of requests is likely to overload the requested service and eventually to cause a trigger for limited availability of resources, this overload trigger is sent prior to any hard resource limitations and to initiate smooth overload control and potential redistribution of the service across neighboring nodes.

6) Service consumption trigger: The resource consumption of individual services needs to be monitored. Based on the type of the service and its resource consumption, a trigger is generated if the service consumption is under or above predetermined levels. Note that the previously-discussed resource availability trigger deals with the aggregate system level resources whereas this trigger copes with the performance of a single service.

B. Grouping Attributes

The key mobility triggers for slice mobility are not alone sufficient to efficiently manage the slice mobility patterns. In addition to triggers, we need means to group various relevant objects (e.g. services, users, and network functions) so the mobility of the slices of resources consumed by those groups can be separately and efficiently managed. Next, we will investigate various grouping attributes to support slice mobility patterns.

1) Grouping By User Subscription Type: Grouping users by using their distinctive subscription types are one of the simplest grouping attributes. In 5G, this identifier is called subscription permanent identifier (SUPI) and it is globally unique throughout the 3GPP system [15]. Another useful identifier in 5G is generic public subscription identifier (GPSI) that identifies a 3GPP subscription for different data networks. With these subscription identifiers, we can separate for example IoT users of a given service provider from mobile broadband users of the same or different providers, or prepaid users from enterprise users. Because this grouping is very coarse, it will be performed in combination with other types of groupings.

2) Grouping By Access Type: Users can be grouped by the access type they use. Access types could be 3GGP (e.g. 4G, 5G, 5G small cell) or non-3GPP (e.g. WiFi and Wi-Max) accesses or multi-access. This grouping attribute is coarse and often needs to be complemented with other grouping attributes.

3) Grouping By Network Slice Type: 3GPP defines three standard types of slices:

- eMBB (enhanced Mobile Broadband) slice,

- URLLC (ultra- reliable low latency communications) slice,
- MIoT (massive IoT) slice.

A UE can be connected at the same time to multiple slices. A slice is associated with a slice identifier, called network slice selection assistance information (NSSAI) that contains information about the slice type [15]. When a UE connects to the 5G network, it will use this identifier to express which slice it wants to join. NSSAI is also used in binding services and 5G network functions to a particular slice. It is very critical for slice mobility to identify that particular slice that will be modified or moved as it impacts multiple bindings between multiple entities. Unfortunately, NSSAI is unique only within one operator domain which introduces additional complexity when a slice is moving across operator boundaries.

4) Grouping By Service Area: Grouping of the consumed resources only is not enough because service area and service availability limitations may exist. A trivial example would be the case of a user "A" that is consuming an ultra-low latency service. If user "A" is a static user or his mobility pattern is restricted to a single service area, this user cannot be grouped with other users that are allowed to use the low latency service in other service areas.

5) Grouping By Access Characteristics: Grouping users by their experienced access characteristics constitute one of the essential groupings attributes to identify a slice mobility pattern. Access characteristics take into account the radio metrics of the access, throughput, and frequency of handovers. For example, in our third use case where passengers are traveling in a high-speed train and receive video service over a network slice, one useful grouping of the users would be based on users experiencing same radio characteristics, same radio frequencies, same handover frequency and using same access points in addition to receiving the same service.

6) Grouping By Geographical Location: Geographical Location is an obvious grouping method that is applicable to all above-mentioned used cases. Geographical location-based grouping is often combined with the other grouping methods, particularly with the mobility pattern grouping. In Geographical Location-based grouping, users, services or slices are classified based on their current location.

C. Slice Mobility Patterns

We classify slice mobility patterns into the following categories: i) full slice mobility; ii) partial slice mobility which includes slice breathing, slice splitting, and slice merging; and iii) slice mobility optimizer which contains slice shrinking pattern. In the remainder of this section, each slice mobility pattern is described, highlighting its respective triggers. As discussed earlier, slice mobility events are generally triggered by the mobility of a group of end-users, the availability of the needed resources in the cloud or by the security aspects of the requested services.

1) Full Slice Mobility: In this use case, we consider a group of users moving to a different location; e.g., a swarm of UAVs moving from the service area of an edge cloud 1 to the service area of another edge cloud 3 (Fig. 1). This group mobility will trigger a service migration process of the entire services and associated resources used by this group. This leads to the notion of full slice mobility as all resources and the services of a slice are impacted by the mobility and are migrated from the original resource location to a new one. Several triggers may be used to indicate the need for this type of mobility. The most important ones are group mobility, limited resource availability, and reliability triggers.

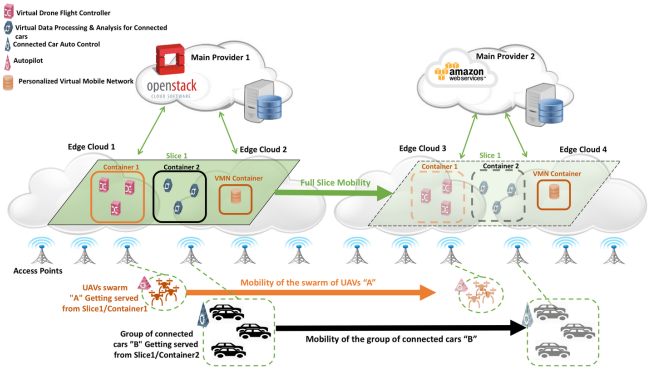


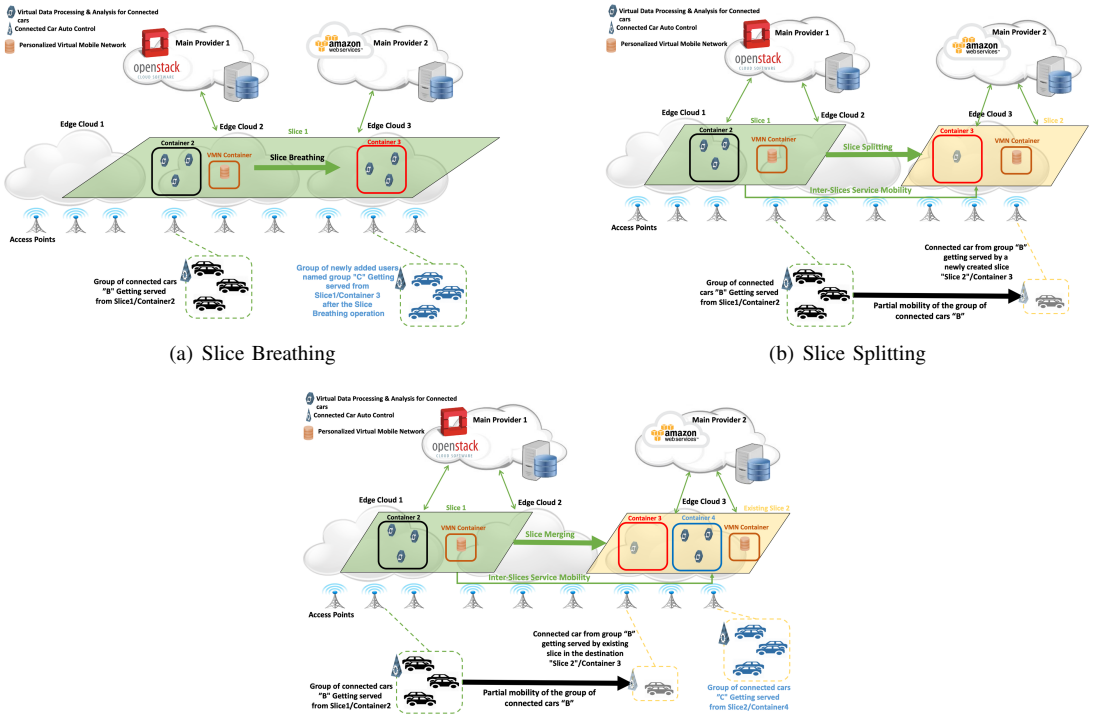
Fig. 1. Full Slice Mobility Pattern

2) Partial Slice Mobility: Full slice mobility is an expensive operation but some of the mobility patterns can be supported with less tedious operations. In partial mobility, only some identified resources of a slice are to be migrated. Thus, two possible cases results, either the network slice will be extended to allow more coverage (Slice Breathing) or a different network slice will be considered in the destination. Concerning the second situation, we consider two cases: i) either existing slices; or ii) newly- created slices. Both of these divergent cases yield two types of slice mobility, namely Slice Splitting and Slice Merging.

a) Slice Breathing: Another use case, depicted in Fig. 2(a), when the group of users "C" attached to the existing slice (slice 1) causing skewed resource consumption, over-consuming a subset of the offered services of the slice 1. A slice breathing operation will be triggered, causing replication of the content of highly-loaded micro-services (e.g. containers), followed immediately by user redirection to this newly-created (sub-)slice to guarantee seamless continuity of the services.

The slice breathing operation can be based on several triggers. Clearly, group mobility trigger, resource availability trigger, service overload trigger, and service consumption trigger can be used to initiate a slice breathing mobility pattern. To summarize, in the slice breathing operation, a slice is temporarily expanded by combining service replication and service migration processes with redirection of users to another slice to match a sudden need for scaling a slice.

In the case of unworkability of slice breathing operations, slice splitting and slice merging are adopted depending on the



(a) Slice Breathing

(b) Slice Splitting

(c) Slice Merging

Fig. 2. Partial Slice Mobility Patterns

situations.

b) Slice Splitting: In the slice splitting case, we opt to create a new slice for the upcoming group of users as shown in Fig. 2(b). This action will be permitted by doing an inter-slice service mobility to ensure the availability of the service in the newly-added slices by exploiting the ability of inter-data-center multiple migrations.

The group mobility trigger, the resource availability trigger, the security trigger, the service overload trigger, and the request overload trigger will be part of the envisaged triggers for the slice splitting use case. However, for these slice mobility patterns, the system should consider the service consumption behavior of end-users as a trigger to choose between slice splitting or slice merging. Effectively, if the requested service is not a delay-sensitive service, the system may opt for a slice splitting action, wait until the creation of the new slice, and start the mobility process by performing migration and replication actions.

c) Slice Merging: On the other hand, considering the availability of interoperability between two slice providers (i.e., two IaaS), the system may optimize the distribution of the new containers forming a set of slices and allowing an inter-slice connectivity. This kind of slice mobility could be very useful in case of no network coverage from the initial slice provider. A multiple migrations process added to a replication process should be envisaged to enable this use case.

The slice merging mechanism presented in Fig. 2(c) is a quite similar mechanism to slice splitting with the exception of the creation phase of the slice because in this case ultra short latency services will be considered, which means that we cannot tolerate the extra time for the slice creation. Due to this constraint, a slice merging mechanism is employed. In case of different slice providers, an interoperability contract is intuitively assumed to be established between these different providers.

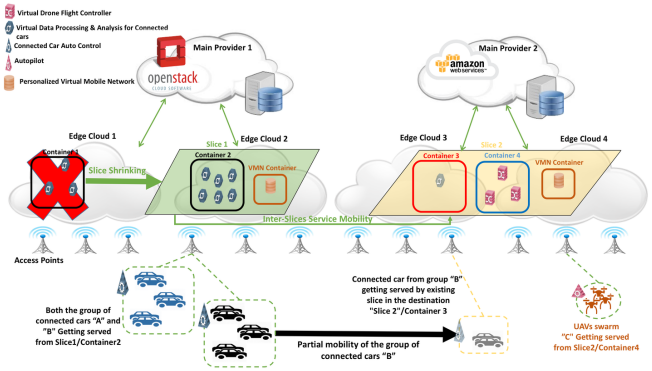


Fig. 3. Slice Shrinking

3) Slice Mobility Optimizer: Slice Mobility Optimizer mechanisms aim for avoiding the waste of unused resources as a result of the mobility of users, a reduced usage of services, and/or the end of the service usage.

a) Slice Shrinking: In Fig. 3, we illustrate the case of a group of users that moved and are served by a different slice (i.e., either existing one or newly created) in their new location (i.e. Edge Cloud 3 & 4). The remaining users are served by Edge Cloud 2, and resources originally used in Edge Cloud 1 will be released, resulting in the shrinkage of the original slice. Slice shrinking is a cleaning mechanism that should follow each case where existing resources are migrated to a new host to release all previous resources that became obsolete during the migration process.

After deciding which slice mobility pattern to execute, the system should run a control algorithm that determines the optimal number of virtualization instances and sets the rules for slice shrinking. For instance, in Fig. 3, we have two containers serving the groups of users A & B. The system can use the number of requests that container 1 or 2 is able to handle. If either of them is able to provide the required service with respect to the SLA terms negotiated before, it can take over the services provided by the other one, that will be ultimately turned off.

Table I summarizes the relationships between the slice mobility patterns and the different mobility triggers. The slice shrinking pattern is not included in the table as it is supposed to be an automatic process activated after each slice mobility pattern to free resources that become unused after resource migration.

TABLE I
RELATIONSHIP BETWEEN KEY ENABLER TRIGGERS & SLICE MOBILITY PATTERNS

Triggers / Slice Mobility	Full Slice Mobility	Slice Splitting	Slice Merging	Slice Breathing
Group Mobility	Yes	Yes	Yes	Yes
Resource Availability	Yes	Yes	Yes	Yes
Reliability	Yes	No	No	No
Security	No	Yes	Yes	No
Service Overload	No	Yes	Yes	Yes
Service Consumption	No	Yes	Yes	Yes

V. ENABLING TECHNOLOGIES & EVALUATION SETUP

This section presents our preliminary evaluation of two key enabling technologies for Slice Mobility: system virtualization with containers and SDN. These technologies allow a fast and seamless allocation of resources to a network slice, while a complete evaluation of the behavior of the migration procedure under different physical setups and user mobility patterns are outside the scope of this work we do evaluate a performance of parallel migrations to better understand its current performance bottlenecks. Our focus on the parallel migration arises from our perceived need to support multiple simultaneous migrations during a slice reconfiguration caused by user mobility across domains.

Fig. 4 portrays the testbed environment envisioned to emulate the slice mobility patterns, targeting the third use case of Section III. In this use case, a slice consists of a set of streaming functions that are used for streaming video

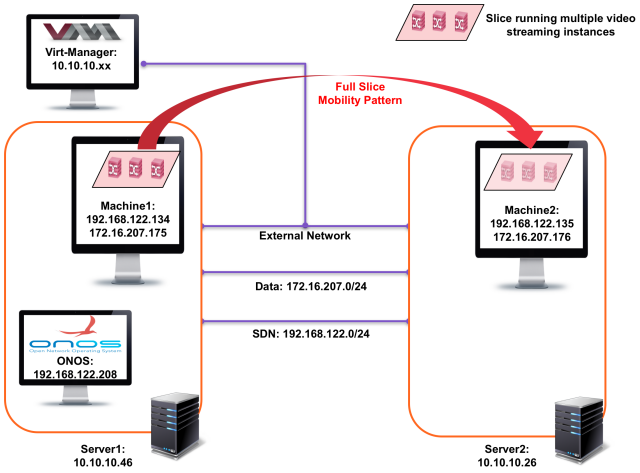


Fig. 4. Testbed setup

services. The testbed consists of two physical servers (i.e., two administrative IaaS domains) as depicted in Fig. 4. Both servers run Ubuntu version server and have KVM as a hypervisor. Server 1 hosts two VMs; one acts as the source cloud for the slice mobility while the other VM acts as an SDN controller that manages the communications between the different servers and their clients. As SDN controller, we used ONOS (Open Network Operating System). However, any other SDN controller could be used as well. The second physical server, server 2, hosts only one VM that plays the role of the destination cloud. Both servers and all VMs are running Ubuntu 16.04 LTS with the 4.4.0-64-generic kernel with 4 CPU cores and 4 GB of memory. The connection between the servers is 1Gbps. An additional host is used for accessing the testbed from an external network.

As previously explained in Section II, the container technology offers better overall performance compared to VMs. The use of system level container, i.e., Linux Container (LXC), technology allows us to support a wider range of applications with the proposed framework. In our environment setup, we, therefore, use the container technology LXC 2.8 and CRIU 2.6. We created three containers; each of them is running an NGINX server to stream videos to three different clients.

In the performance evaluation, we chose the full slice mobility pattern as it is the most complicated and computation-heavy scenario. The migration process triggers the migration of running instances in parallel from the source cloud to the destination cloud. To automate the orchestration process, we built a framework to handle the parallel container migrations. We considered the resource availability trigger to enable the parallel migrations. We used a mix between the grouping by service area method, and the grouping by geographical location method since our clients are static and are served from their respective video streaming servers. By leveraging the SDN paradigm, we created an overlay networking on top of the physical network. Such an action allowed us to design an isolated network topology over multiple physical data centers, and at the same time, carrying out inter-data-center (i.e., inter-IaaS) parallel migrations.

For enabling the full slice mobility, the test is performed using the iterative migration approach [16]. We adopt the pre-copy logic to perform the whole operations of live migration starting with the copy of the disk until the memory copy. Then, the SDN controller handles the path redirection phase and finally the container is restored to that target node.

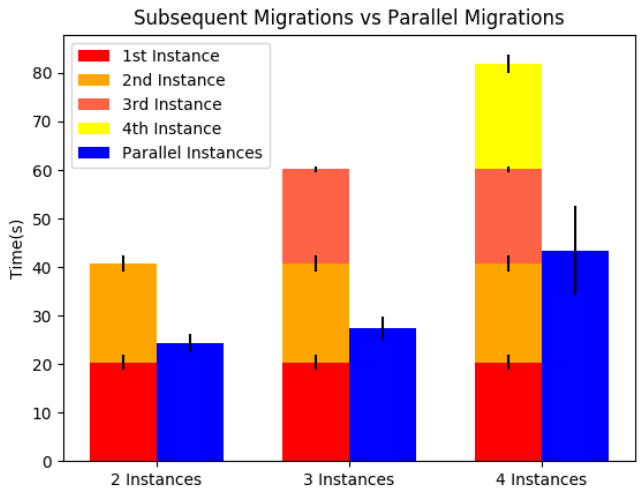


Fig. 5. Subsequent Migrations vs Parallel Migrations

Fig. 5 respectively shows the duration of 2, 3 and 4 subsequent migrations vs the same number, however, leveraging parallel migrations. The test is performed with identical sizes of containers in order to carry out fair tests while comparing the parallel migrations to the subsequent one. In Fig. 5, the X-axis represents the number of LXC migrated containers while the Y-axis represents the migration time in seconds. For each bar (i.e., parallel and sequential migration), we plotted the 95% confidence interval of the mean.

The mean total migration time for two subsequent migrations was $40.7123s$ and the 95% confidence interval for this experiment was $2.0007s$. For the parallel migrations, we obtained a mean total migration time of $24.3023s$ and $1.9764s$ as its 95% confidence interval. Regarding the migration of three instances, we achieved a mean total migration time of $60.1735s$ and $27.3984s$, respectively, for both sequential and parallel migrations. Finally, on the subject of four instances migrations scenario, the mean total migration time is $81.9131s$ for the sequential migrations with 95% confidence interval of $2.6310s$, while these values are $43.4268s$ and $9.0867s$, respectively, for the parallel migrations. The results obtained through this evaluation reveal that the parallel migrations for enabling a prompt slice mobility are more efficient when compared to the legacy sequential migration strategy.

VI. Conclusion and Remarks

The network slicing paradigm unquestionably offers a powerful apparatus to support verticals' services. According to this paradigm, a group of users is associated with dedicated computing, storage, and network resources tailored for a given vertical use case. The use of specialized resources in a slice implies that such resources are not available everywhere in the network but require careful resource allocation policies and control. Therefore, there is a strong need to extend the notion of mobility that is traditionally limited to user devices or services only without much concern of combined resource availability needed for network slicing. With network slicing, the user and service mobility events become more correlated leading to new mobility patterns. In this article, we introduced slice mobility patterns with corresponding grouping methods and relevant mobility triggers.

The proposed mobility patterns leverage MEC to offer ultra-short latency communication infrastructure, SDN for fast flow resumption, live migration to ensure the high-availability of services and the Follow me Edge concept to support end-users' mobility. The proposed framework is validated using a realistic testbed. Interesting results were obtained, implying total time migration evaluation depends on the used video streaming delivery content. The obtained results also demonstrate the dominance of the proposed parallel migrations when compared to the legacy sequential migrations strategy. Based on the achieved results and the presented key enabler triggers in addition to the grouping methods, it can be concluded that a mechanism to select the right combination of techniques to be used for efficiently ensuring a slice mobility pattern action is indispensable. For the authors, a future research direction would be to investigate Artificial Intelligence techniques for smart and cost-efficient triggering and grouping methods for slice mobility patterns.

Acknowledgment

This work was supported in part by the Academy of Finland Project 6Genesis Flagship (grant no. 318927), and in part by the European Unions Horizon 2020 Research and Innovation Program through the MATILDA Project under Grant No. 761898.

References

[1] N. Alliance, "5G white paper," Tech. Rep., February 2015. [Online]. Available: https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf

[2] K. Ramantas, A. Antonopoulos, E. Kartsakli, P. Mekikis, J. Vardakas, and C. Verikoukis, "A c-ran based 5g platform with a fully virtualized, sdn controlled optical/wireless fronthaul," in *2018 20th International Conference on Transparent Optical Networks (ICTON)*, July 2018, pp. 1–4.

[3] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, May 2017.

[4] I. Afolabi and T. Taleb and K. Samdanis and A. Ksentini and H. Flinck, "Network Slicing; Softwarization: A Survey on Principles, Enabling Technologies; Solutions," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2018.

[5] T. Taleb, B. Mada, M. I. Corici, A. Nakao, and H. Flinck, "Permit: Network slicing for personalized 5g mobile telecommunications," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 88–93, May 2017.

[6] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[7] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[8] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 1350–1354.

[9] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, September 2013.

[10] A. Aissioui, A. Ksentini, A. Gueroui, and T. Taleb, "On enabling 5g automotive systems using follow me edge-cloud concept," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2018.

[11] F. Z. Yousaf, M. Gramaglia, V. Friderikos, B. Gajic, D. von Hugo, B. Sayadi, V. Sciancalepore, and M. R. Crippa, "Network slicing with flexible mobility and QoS/QoE support for 5G Networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 1195–1201.

[12] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *2nd Symposium on Networked Systems Design and Implementation (NSDI 2005), May 2-4, 2005, Boston, Massachusetts, USA, Proceedings.*, 2005. [Online]. Available: http://www.usenix.org/events/nsdi05/tech/clark.html

[13] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman, "Crossroads: Seamless vm mobility across data centers through software defined networking," in *2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 88–96.

[14] Y. C. Tay, K. Gaurav, and P. Karkun, "A performance comparison of containers and virtual machines in workload migration context," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2017, pp. 61–66.

[15] 3GPP, "System Architecture for the 5G System; Stage 2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 03 2018, version 15.1.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[16] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, and H. Flinck, "Towards a Fast Service Migration in 5G," in *IEEE Conference on Standards for Communications and Networking, IEEE CSCN*, Forthcoming, Oct 2018.