MESON: Optimized Cross-Slice Communication for Edge Computing

George Papathanail, Angelos Pentelas, Ioakeim Fotoglou, Panagiotis Papadimitriou, Konstantinos V. Katsaros, Vasileios Theodorou, Sergios Soursos, Dimitrios Spatharakis, Ioannis Dimolitsas, Marios Avgeris, Dimitrios Dechouniotis, and Symeon Papavassiliou

Abstract

Network slicing is set out to address crucial needs of 5G, including support for multi-service provisioning. Focusing on resource and performance isolation, as well as security concerns associated with multi-tenancy, existing management and orchestration (MANO) frameworks typically offer network slices in the form of isolated bundles of computing, storage, and network resources across the network infrastructure, including the edge. However, network slice instantiation in its prevailing form raises significant concerns related to performance and resource utilization, hindering potential business-to-business (B2B) synergies. In this article, we discuss a new aspect of network slicing, optimized cross-slice communication (CSC). We argue that multi-tenancy and service co-location open up unique opportunities for B2B interactions, inter-service communications, and service composition, especially in the domain of edge computing and location-based services. In this context, we present optiMized Edge Slice OrchestratioN (MESON), a MANO-based architecture for optimized CSC in edge clouds. We discuss the main architecture components and descriptors, as well as the steps required for the discovery of CSC-enabled services and the establishment of optimized CSC among co-located slices.

Introduction

Over the last years, there has been an increasing interest in network slicing, with various application domains, such as cellular networks and edge computing. Network slicing, enabled by network function virtualization (NFV) [1–3] and software-defined networking (SDN) [4], essentially allows network providers to lease bundles of computing, storage, and network resources to service providers, active within commercial domains often referred to as verticals. This opens up a unique opportunity for services to get deeply integrated into the (edge of the) network infrastructure, realizing innovative applications with stringent performance requirements in various domains of social and economic activity (e.g., automotive, media, e-health, manufacturing).

Taking a step further, and in an analogy to the well-established cloud realm, we expect these technological advances to facilitate the emergence of a broader service ecosystem, allowing cross-service interactions. Such interactions may significantly vary, from existing operations, such as single sign-on systems [5], to currently emerging function-as-a-service (FaaS) deployments [6], to future service interactions in the context of location-based (edge) services (e.g., tourist guide applications integrating advertisement or social network information) [7].

The prevailing way of slice instantiation inhibits optimized cross-slice communication (CSC), even if the slices are located in the same data center, rack, or server. More precisely, the communication path between co-located slices is significantly stretched, since all traffic must leave and re-enter the data center after traversing the (mobile) network core. This is more crucial for edge computing, because of the larger distance between the edge cloud infrastructures and the network core. As such, potential cross-service interactions would experience higher latency and increased expenditure, since additional bandwidth is consumed from the backhaul and transport network.

To overcome this limitation, we present a new management and orchestration (MANO)-based architecture, namely optiMized Edge Slice OrchestratioN (MESON), for optimized and secure CSC within edge clouds, fostering cross-slice/tenant interactions for next-generation services. MESON provides the necessary means for the establishment of cross-service interactions, exploiting opportunities for service co-location. In particular, MESON encompasses orchestration primitives for CSC-enabled service advertisement, service discovery, and CSC establishment, subject to operations/business support system (OSS/BSS)-level procedures expressing the intent of service providers to establish synergy. To this end, we define multi-access edge computing (MEC)-based descriptors for the expression of CSC offered services and intents, and we also explain the interactions between the MESON architecture components for CSC establishment.

The authors discuss a new aspect of network slicing, optimized cross-slice communication (CSC). They argue that multi-tenancy and service co-location open up unique opportunities for B2B interactions, inter-service communications, and service composition, especially in the domain of edge computing and location-based services.

Cross-Slice Communication

Network slice provisioning is commonly mandated by isolation, which, in a strict form, hinders communication optimizations between network slices and the corresponding service components in the event of service co-location. For instance,

George Papathanail, Angelos Pentelas, Ioakeim Fotoglou, and Panagiotis Papadimitriou are with the University of Macedonia; Konstantinos V. Katsaros, Vasileios Theodorou, and Sergios Soursos are with Intracom Telecom; Dimitrios Spatharakis, Ioannis Dimolitsas, Marios Avgeris, Dimitrios Dechouniotis, and Symeon Papavassiliou are with National Technical University of Athens.

Digital Object Identifier: 10.1109/MCOM.001.2000207

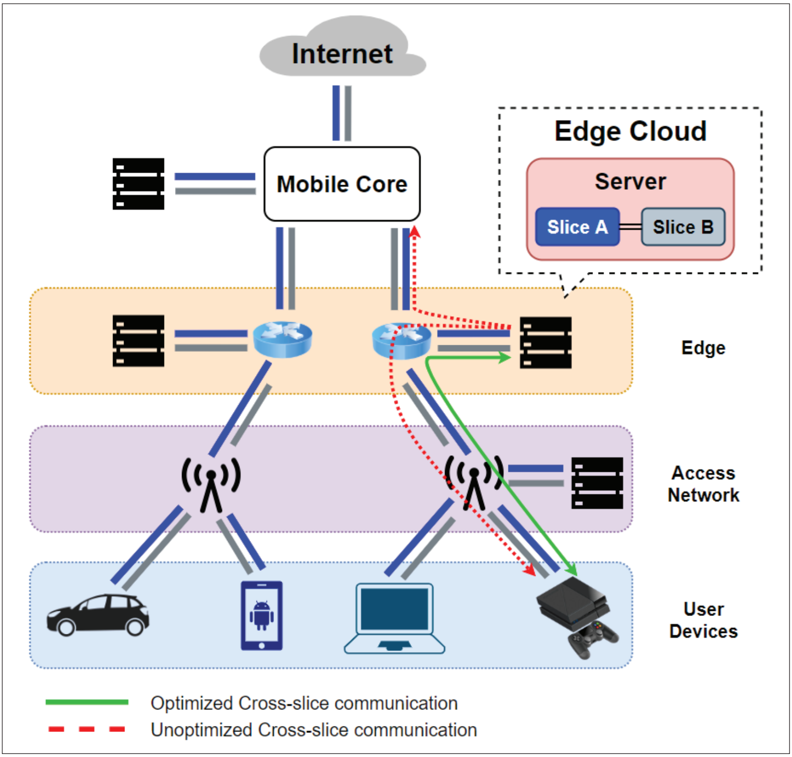


Figure 1. Optimized vs. unoptimized cross-slice communication.

a video-on-demand (VoD) service may need to combine the playback of a video file with an overlaid personalized online advertisement offered by a third-party service provider.

Figure 1 highlights this limitation in terms of CSC establishment. The red dotted line illustrates the communication path between two co-located (but isolated) slices. Stemming from the need for resource/traffic isolation, these two slices will be realized as entirely distinct networks. As such, routing policies will direct the traffic from one slice outside of the data center (DC), through the (mobile) core network, and back to the same DC in order to reach the other slice.

Nevertheless, the shared nature of the (edge) cloud infrastructure opens up opportunities for shortcuts in this communication. Therefore, we promote a form of direct peering between the two slices, which we call *optimized CSC* (represented by the green line in Fig. 1). In this case, CSC traffic is confined within the DC that hosts the two slices, obviating the need to traverse the entire mobile network infrastructure. The traversal of CSC traffic over short paths is very critical when considering the focus of edge computing on the support of low-latency applications. Besides latency savings, optimized CSC is further expected to reduce the traffic volume in the backhaul/transport network.

In the following, we provide a schematic high-level illustration of the potential latency gains from optimized CSC. To this end, we deploy two virtual machines (VMs) in an edge cloud infrastructure (Stackpath [8]). These VMs correspond to service components of two different (but co-located) network slices. In addition, we deploy a third VM in the same region, using Amazon EC2. This VM essentially acts as a (mobile) core gateway, which routes traffic between the two other VMs in the case of unoptimized CSC (at which traffic is enforced to leave the edge cloud). Note that there is no computation performed by this third VM.

Figure 2 illustrates the latency measured with optimized and unoptimized CSC. Although in both cases latency appears to be low, for delay-sensitive applications the advantage of optimized CSC is significant. We specifically observe that for 80 percent of the delay measurements, optimized CSC yields a latency saving of approximately up to 70 percent. We further note that unoptimized CSC tends to skew the tail of the latency distribution, as opposed to optimized CSC, at which the latency is bounded at approximately 3 ms. We hereby stress the fact that for time-critical applications (typical case for edge computing), low latency is not only important on an average basis but also on its upper bound. In this respect, unoptimized CSC cannot comply with such stringent delay budgets, which may create serious implications (e.g., safety could be compromised in automotive applications). Consequently, the lower delays incurred by CSC traffic (in the case of optimized CSC) are critical for the achievement of service key performance indicators (KPIs) in cross-service interactions.

MANO Primer

Since the proposed CSC orchestration framework is based on European Telecommunications Standards Institute (ETSI) NFV MANO, we provide background information on MANO and the associated descriptors. MANO provides management and orchestration functionality across two layers:
- The virtualized infrastructure manager (VIM), which, as its name implies, is responsible for the unified management of the physical infrastructure across all resource types (i.e., compute, storage, network)
- The *orchestration* layer, which encompasses modules for service and resource orchestration, such as life cycle and virtual network function (VNF) state management

A widely used MANO platform is OSM [9], which commonly runs on top of OpenStack [10].

We also briefly discuss the main constructs of MANO's information model. MANO uses the VNF as the lowest structural element that manages and orchestrates, specified by its descriptor (i.e., `vnfd`). In essence, the `vnfd` is a configuration template which includes attributes, such as the virtualized resource requirements of the respective VNF. MANO further defines network services, as a composition of VNFs. To this end, MANO utilizes the network service descriptor (`nsd`), which is also a configuration template. As such, an `nsd` is associated with `vnfds` along with the virtual links that connect them. In other words, the `nsd` specifies the service graph. Finally, the network slice template (`nst`) is the top-level construct that describes a network slice, which, in turn, encompasses a set of inter-connected network services. Among various attributes, the `nst` can be associated with a particular service or a traffic class in the context of 5G, such as enhanced mobile broadband, ultra-reliable low-latency communications, or massive Internet of Things (IoT).

MESON Architecture

We now discuss the proposed MANO-based architecture, which facilitates the establishment of optimized communication between network slices,

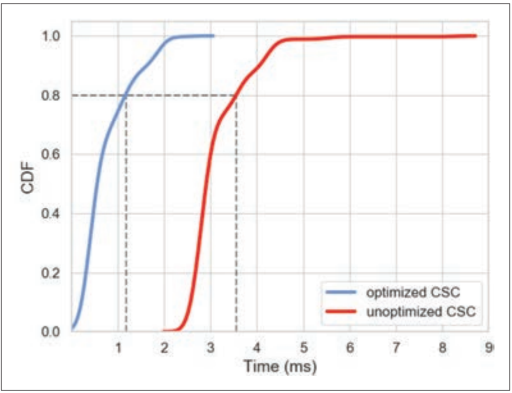


Figure 2. CDF of measured delay with optimized and unoptimized CSC.

co-located in the same edge cloud infrastructure. MESON is currently applicable to a single edge cloud provider with multiple points of presence (PoPs). The MESON architecture has been specified with the following business roles in mind.

The Infrastructure Provider owns the physical cloud infrastructure and offers resources under lease basis for slice deployment.

The CSC-Enabled Service Provider (CSC-SP) is the tenant of a network slice instantiated by the infrastructure provider. The CSC-SP also offers CSC-enabled services in selected PoPs to other slice tenants, which we call CSC-enabled Service Consumers.

The CSC-Enabled Service Consumer (CSC-SC) is a slice tenant that wishes to establish communication with another slice, co-located in the same edge cloud, by means of optimized CSC.

Architecture Overview

Within the context of MESON, the realization of CSC raises the following key functional requirements:

- The advertisement of CSC offerings by a CSC-SP
- The expression of interest from a CSC-SC for the establishment of communication with a CSC-SP
- The binding of CSC interests with offerings for CSC service consumption
- Communication path setup for CSC establishment between the slices of the CSC-SP and CSC-SC

In order to meet these requirements, we have specified a three-layer MANO-based architecture, as depicted in Fig. 3. The two lower layers essentially encompass the management and service/resource orchestration functionality of MANO.

The main functionality of MESON resides on the top layer, including the following components.

The Service Registry handles the binding of CSC interests with CSC offerings, subject to an exposed policy, which we discuss in the next section. The Service Registry maintains a list of service instances offered by CSC-SPs. The interests expressed by CSC-SCs are also directed to the Registry.

PoP Selection identifies the most suitable PoP for the deployment of a slice requested by a CSC-SC. This module generates a PoP ranking, based on *hard* and *soft* requirements from the slice tenant. The hard requirements are associated with attributes, such as the location (availability zone) of the PoP. The soft requirements are associated with computing and network performance indicators (e.g., service response time), cost parameters, and technical support. The above criteria can be expressed by various types of numeric values, such as binary, range values, and unordered sets. Therefore, a multi-criteria decision making (MCDM) approach is deemed suitable for the PoP ranking [11].

The MESON Agent comprises the client-facing interface for MESON. Both CSC service advertisements and interests are received by MESON via this agent.

Service Discovery

Service discovery comprises a prerequisite for CSC establishment, as it performs the binding of CSC interests with service offerings from CSC-SPs. This process is handled by the Service Registry. In order to convey the required information from CSC-SPs/CSC-SCs (via OSS/BSS) to the Service Registry, we extend the existing descriptors provided by ETSI MEC specifications [12]. As shown in Fig. 4, MESON employs the MEC application descriptors (`AppD`) for the description of service attributes. The registration of a CSC service is established through the `appServiceProduced` field, which is of type `ServiceDescriptor`. In addition, a CSC interest is expressed through the `appServiceRequired` field, which is of type `ServiceDependency`.

We further extend the `ServiceDescriptor` type with a `PeerPolicy` field used to describe the policy of CSC-SP regarding the CSC establishment. A peering policy contains a list of policy-related properties, such as the highest acceptable computing and network resources (described as a new `DependencyProfile` type) and the scale-out scheme (if any) supported by the CSC-SP in response to the increased load due to the CSC traffic. On the request side, we extend the `ServiceDependency` type with a `DependencyProfile` field describing the expected load of the intended CSC, as well as the desired latency of the communication path between the corresponding CSC counterparts.

Concerning the binding process, the Service Registry employs a mechanism that matches CSC interests with registered service instances in order to identify the various CSC establishment options. This matching is based on the service type (as expressed within the `ServiceDescriptor` and `ServiceDependency`) and the CSC peering policy (as described within the `PeerPolicy` and `DependencyProfile`).

The binding relies on a filtering feature that discards any slice instances which do not comply with the CSC criteria.

CSC Setup

To achieve secure, efficient, and controlled communication among different slices, we introduce the notion of a dedicated CSC slice (i.e., an intermediate network entity, operated by the infrastructure provider) purposed for CSC and policy control (Fig. 5). A CSC slice can incorporate additional functionalities, such as (deep) packet inspection, monitoring, and billing. Besides policy enforcement and control, a dedicated slice for CSC facilitates the isolation of CSC traffic from other traffic flow-

Network slice provisioning is commonly mandated by isolation, which, in a strict form, hinders communication optimizations between network slices and the corresponding service components in the event of service co-location. For instance, a video-on-demand service may need to combine the playback of a video file with an overlaid personalized online advertisement offered by a third-party service provider.

To achieve secure, efficient, and controlled communication among different slices, we introduce the notion of a dedicated CSC slice, an intermediate network entity purposed for CSC and policy control. A CSC slice can incorporate additional functionalities, such as (deep) packet inspection, monitoring, and billing

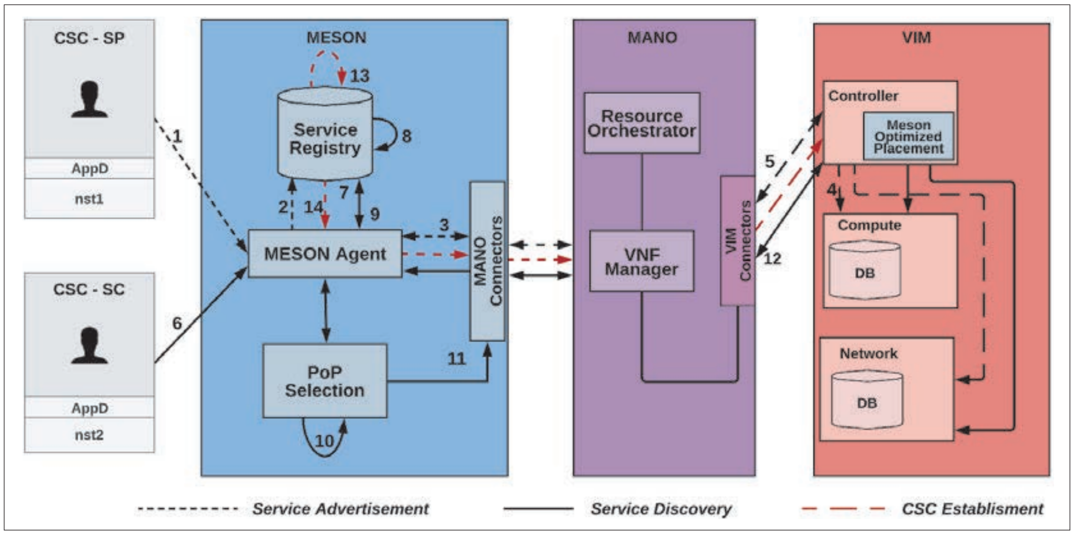


Figure 3. MESON architecture and workflows.

ing in the edge DC. The configuration of the CSC slice requires former knowledge of the connection points of the two slices that seek to communicate, as well as the PoP that hosts them. Such information has been previously stored in the Service Registry during the instantiation of each slice. In MESON, we consider several types of CSC slice templates stored in the Service Registry. This emphasizes the uniqueness of each CSC, as well as the need to address specific requirements that may occur according to the peering policies.

CSC Establishment

We now discuss the steps taken by MESON for CSC establishment, which is carried out as a sequence of the following three phases:

- Service advertisement
- Service discovery
- CSC slice instantiation

For brevity, in the illustration of this workflow (Fig. 3), we refer to the `nst`, which encompasses all descriptors that comprise the slice (i.e., `vnfd`, `nsd`).

During the initial phase, the CSC-SP conveys the `nst` and `AppD` to the *MESON Agent* (***step 1***), which, in turn, forwards the `AppD` to the Service Registry (***step 2***). Since the instantiation of CSC-SP's slice is not dependent on other running slices (i.e., the deployment of the two communicating slices is handled by MESON in an asynchronous manner), the MESON Agent directs the respective `nst` to the MANO layer for slice deployment (***step 3***). Subsequently (***step 4***), the slice is assigned to a PoP, where it is deployed by the respective VIM. Furthermore, the slice is associated with a connection point, which is exposed for the purpose of optimized communication with another slice. This information is inserted into the Service Registry (***step 5***).

The second phase commences with the submission of the CSC interest from the CSC-SC. This is expressed via the `AppD` and `nst`, which are submitted to the *MESON Agent* (***step 6***). Since MESON seeks the co-location of communicating slices, the placement of the CSC-SC's slice will depend on the location (i.e., PoP) of the CSC-SP's slice. As such, a lookup is performed in the Service Registry (***step 7***), based on the CSC-SC's `AppD`. This process is illustrated in ***step 8***. The outcome of this matching is conveyed to the *PoP Selection* module (***step 9***). The latter ranks the different CSC options with respect to KPIs advertised from the PoPs. Eventually, MESON applies an MCDM method for computing the final ranking of the PoP candidates (***step 10***) [11]. This concludes the configuration of the instantiation parameters of the CSC-SC's slice, thus enabling its deployment (***step 11***). In the final step (***step 12***) of this phase, the exposed connection points (at which the CSC slice will be attached) are stored in the Service Registry.

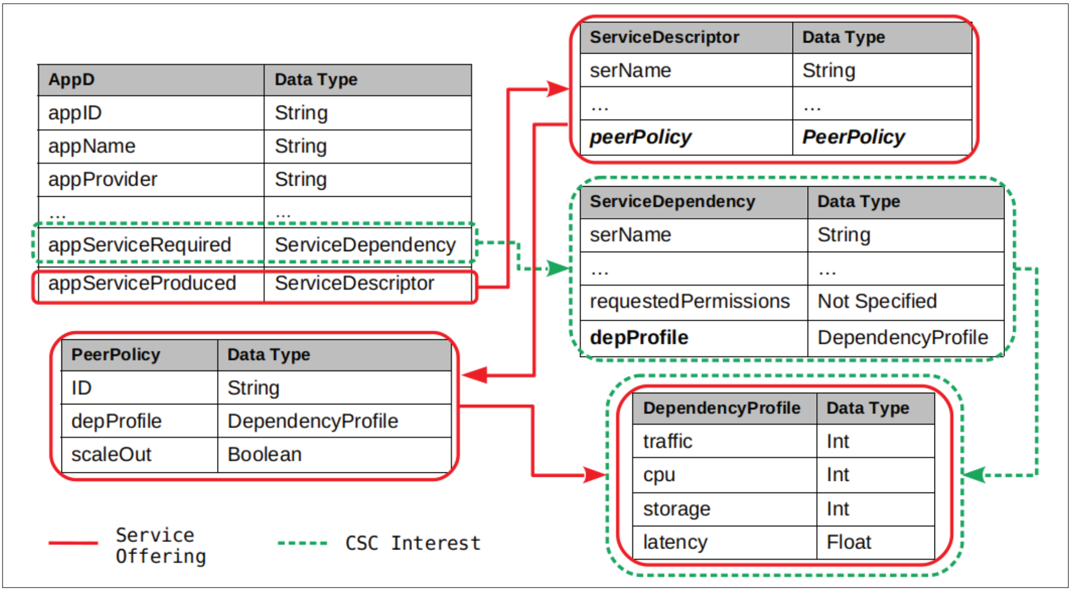
The third and final phase consists of the instantiation of the intermediate CSC slice. In this respect, ***step 13*** represents the configuration of the intermediate slice according to the above parameters. Finally, in ***step 14*** the generated `nst` is conveyed to the lower layers, enabling the VIM to instantiate the CSC slice and perform its binding with the slices of the CSC-SP and CSC-SC.

Experimental Evaluation

In this section, we present experimental results from an initial prototype implementation of the MESON architecture. Specifically, our implementation includes the Service Registry, the PoP Selection module, and the MESON Agent. All components are executed within a VM running unmodified Ubuntu 18.04 with 4 vCPUs and 4 GB of main memory. Communication among the three MESON components is achieved through REST application programming interfaces (APIs). For the implementation of the APIs, we use Python and Flask. We employ the no-SQL MongoDB in order to store and index the uploaded AppD files in the Service Registry. The PoP selection module has been implemented using structured SQLite.

Our experimentation is focused on CSC service discovery and PoP selection, which are triggered upon the expression of CSC interest from a CSC-SC. We assume that the Service Registry already contains a sufficient number of service instances. In this respect, we measure the execution time of service discovery and PoP selection.

Our tests are executed in a range of 10 to 100 PoPs, which correspond to diverse footprints of edge cloud providers. For each PoP, we populate



AppD	Data Type
appID	String
appName	String
appProvider	String
...	...
appServiceRequired	ServiceDependency
appServiceProduced	ServiceDescriptor

ServiceDescriptor	Data Type
serName	String
...	...
peerPolicy	***PeerPolicy***

ServiceDependency	Data Type
serName	String
...	...
requestedPermissions	Not Specified
depProfile	DependencyProfile

PeerPolicy	Data Type
ID	String
depProfile	DependencyProfile
scaleOut	Boolean

DependencyProfile	Data Type
traffic	Int
cpu	Int
storage	Int
latency	Float

Figure 4. Descriptors for CSC interests and service offerings.

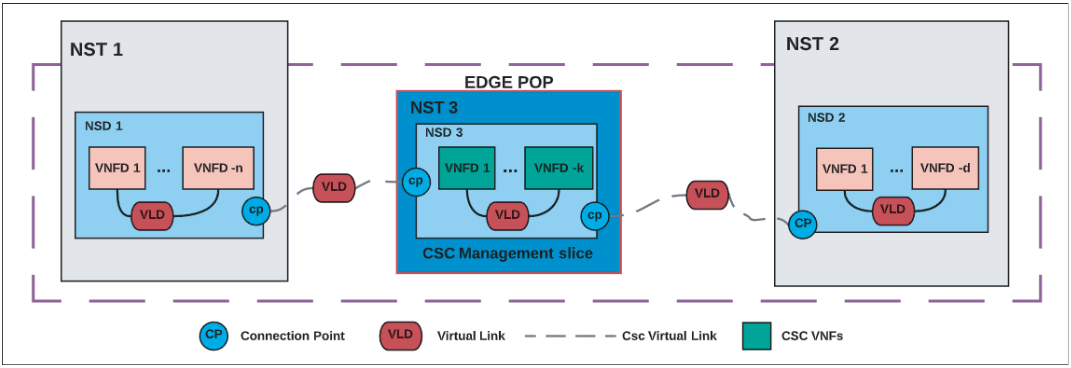


Figure 5. Intermediate CSC slice attached to connection points of communicating slices.

Our experimentation is focused on CSC service discovery and PoP selection, which are triggered upon the expression of CSC interest from a CSC-SC. We assume that the Service Registry already contains a sufficient number of service instances. In this respect, we measure the execution time of service discovery and PoP selection.

100 service instances (in the form of `AppDs`) in the Service Registry (e.g., in the case of 100 PoPs, the Registry contains 10K `AppDs`). As such, both the Service Registry and the PoP selection module are evaluated within a comparable data range, since the Registry deals with `AppDs`, whereas the PoP selection component accounts only for PoPs.

Figure 6 illustrates the execution time for service discovery and PoP selection. The delay incurred during service discovery increases linearly with the number of PoPs and does not exceed 15 ms (for 100 PoPs and 10k `AppDs`). The time required for PoP selection yields a polynomial-like increase with the number of PoPs. This stems from the MCDM mechanism utilized by the PoP selection module, whose complexity increases in a similar manner. Consequently, both the service discovery and PoP selection processes can be completed in the range of tens of milliseconds, even for edge cloud providers with a number of PoPs as high as 100. This indicates that the operations carried out in the MESON layer do not introduce any significant performance overhead in terms of CSC establishment.

Use Cases

Video Streaming. Video streaming occupies the lion's share of today's total Internet traffic. According to the latest Cisco Visual Networking Index (VNI) Complete Forecast, video streaming represents 67 percent of all Internet traffic, and it is projected to increase up to 80 percent by 2021. As expected, this puts not only the content providers, but the network operators as well, under severe stress. To alleviate this load, popular techniques such as content delivery networks (CDNs) have been utilized in the current network infrastructure. However, 5G brings even further possibilities, by allowing network services to be offered as VNFs deployed on DCs across the (edge of the) network infrastructure. Taking advantage of the new 5G features, network operators can deploy their own (virtual) CDN services (vCDN) much closer to the end user. In such an ecosystem, one can imagine multiple virtual network operators (VNOs) offering CDN services that are co-located in the same DC. Hence, VNOs can benefit from service-based interactions, offering and consuming video streaming services (storage, compression, transcoding, etc.) to/from other VNOs. In this as-a-service model, streaming services can be exchanged via direct communication of corresponding co-located VNFs, whereas VNOs can benefit from additional revenue streams and load balancing.

Industry 4.0. One of the prerequisites, but open challenges, for Industry 4.0 is the physical interaction between robots that operate on the factory floor and seek to execute a specific task in a coordinated and secure way. Robotic pro-

Additional performance tests using OSM and OpenStack indicate that a CSC slice with a set of management and security functions can be instantiated on the order of seconds. As future work, we plan to quantify the gains of optimized CSC using service-level performance indicators, such as service response time.

cesses (e.g., localization [13]) rely on various sensors and complex algorithms that require plenty of computing resources. Despite the recent boost in the computing capabilities of robotic systems, local execution still remains a time-consuming process. For this reason, current trends in network service delivery dictate that robotic applications are being developed and treated as VNFs, making their deployment to the cloud feasible. In such a setting, autonomous robotic clusters, deployed by different vendors on the same factory floor, can communicate through their corresponding slices in order to coordinate and perform their tasks (e.g., inventory loading/unloading) more efficiently.

Conclusions

The increasing interest in cross-service/tenant interactions in edge clouds raises the need for optimized CSC and mechanisms for its orchestration. To this end, we present the architecture design of a MANO-based CSC orchestration framework and exemplify the required interactions for the discovery of CSC-enabled services and the establishment of optimized CSC between co-located slices. We further present evaluation results from an initial prototype implementation of the MESON architecture. Our evaluations show that the main operations in the MESON layer (i.e., service discovery, PoP selection) incur low delays (less than 40 ms with a number of PoPs as high as 100). Additional performance tests using OSM and OpenStack indicate that a CSC slice with a set of management and security functions can be instantiated on the order of seconds. As future work, we plan to quantify the gains of optimized CSC using service-level performance indicators, such as service response time.

References

[1] NFV White Paper, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action, Issue 1," Oct. 2012.

[2] M. Kourtis *et al.*, "T-nova: An Open-Source Mano Stack for NFV Infrastructures," *IEEE Trans. Network and Service Management*, vol. 14, no. 3, Sept. 2017, pp. 586–602.

[3] R. Mijumbi *et al.*, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 1, 2015, pp. 236–62.

[4] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," *SIGCOMM Comp. Commun. Rev.*, vol. 44, no. 2, Apr. 2014, pp. 87–98.

[5] A. Pashalidis and C. J. Mitchell, "A Taxonomy of Single Sign-On Systems," *Info. Security and Privacy, 8th Australasian Conf.*. Springer-Verlag, 2003, pp. 249–64.

[6] M. Villamizar *et al.*, "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," *CCGrid*, 2016.

[7] H. Gao *et al.*, "Exploring Temporal Effects for Location Recommendation on Location-Based Social Networks," *RecSys '13*, 2013, pp. 93–100.

[8] "Stackpath — Secure Edge Computing"; http://stackpath.com, accessed Dec. 1, 2019.

[9] OSM; http://osm.etsi.org.

[10] Openstack; http://www.openstack.org.

[11] Dechouniotis *et al.*, "Fuzzy Multi-Criteria Based Trust Management in Heterogeneous Federated Future Internet Testbeds," *Future Internet*, vol. 10, no. 7, 2018, p. 58.

[12] ETSI, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application Life Cycle, Rules and Requirements Management," 2017, pp. 19–28; https://www.etsi.org/deliver/etsi gs/mec/001 099/01002/01.01.01 60/gs mec01002v010101p.pdf.

[13] M. Avgeris *et al.*, "Single Vision-Based Self-Localization for Autonomous Robotic Agents," *FiCloudW*, 2019.

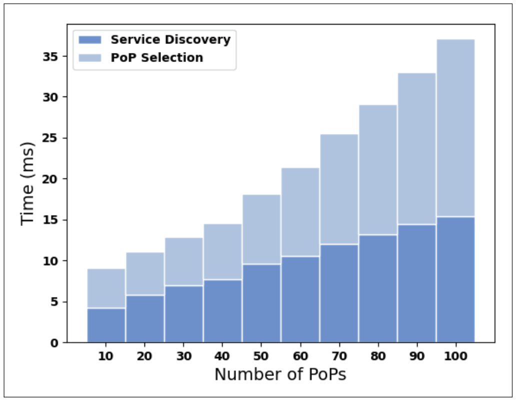


Figure 6. Execution time of service discovery and PoP selection.

Biographies

George Papathanail (papathanail@uom.edu.gr) is a Ph.D. candidate in the Department of Applied Informatics at the University of Macedonia, Greece. His research interests include network slicing, NFV, and cloud computing.

Angelos Pentelas (apentelas@uom.edu.gr) is a Ph.D. candidate in the Department of Applied Informatics at the University of Macedonia. His research interests include discrete optimization, NFV, and network management.

Ioakeim Fotoglou (fotoglou@uom.edu.gr) is a Ph.D. candidate in the Department of Applied Informatics at the University of Macedonia. His research interests include network and cloud security, and 5G architectures.

Panagiotis Papadimitriou (papadimitriou@uom.edu.gr) is an assistant professor in the Department of Applied Informatics at the University of Macedonia. His research activities include Internet architectures, network processing, SDN, data center networking, and edge computing.

Konstantinos Katsaros (konkat@intracom-telecom.com) is a senior R&D engineer at Intracom S.A. Telecom Solutions, Greece, working on edge computing and NFV/MANO for 5G networks. He was a research associate at Telecom ParisTech, France, and subsequently at University College London, United Kingdom.

Vasileios Theodorou (theovas@intracom-telecom.com) is an R&D engineer at Intracom S.A. Telecom Solutions, working on NFV and edge computing. He was a research associate at the Polytechnic University of Catalonia and a research assistant at York University of Toronto, Canada.

Sergios Soursos (souse@intracom-telecom.com) is a master R&D engineer at Intracom S.A. Telecom Solutions, working on 5G networks, IoT interoperability, and cloud/edge computing. He has also worked on cloud traffic management, ICN, peer-to-peer networks, and overlay network management.

Dimitrios Spatharakis (dspatharakis@netmode.ntua.gr) is a Ph.D. candidate in the School of Electrical and Computer Engineering (ECE) at National Technical University of Athens. His research interests include cyber physical systems, IoT, NFV/SDN, and edge computing.

Ioannis Dimolitsas (jdimol@netmode.ntua.gr) is a Ph.D. candidate in the School of ECE at National Technical University of Athens. His research interests include multi-criteria optimization, fedetated clouds and NFV/SDN.

Marios Avgeris (mavgeris@netmode.ntua.gr) is a Ph.D. candidate in the School of ECE at National Technical University of Athens. His research interests include control theory, cloud computing, IoT, semantic web technologies, and network monitoring.

Dimitrios Dechouniotis (ddechou@netmode.ntua.gr) is a senior researcher in the School of ECE at National Technical University of Athens. His research interests include control theory, cloud computing, and IoT.

Symeon Papavassiliou (papavass@mail.ntua.gr) is a professor in the School of ECE at National Technical University of Athens. He has an established record of publications in the field of next generation network modeling, optimization, and management, with more than 300 technical journal and conference papers.