The Journal of Systems and Software 152 (2019) 24–31

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Latency-aware Virtualized Network Function provisioning for distributed edge clouds

Jungmin Son*, Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, VIC 3010, Australia

ARTICLE INFO

Article history:
Received 7 November 2018
Accepted 18 February 2019
Available online 22 February 2019

Keywords:
Cloud computing
Edge computing
Network Function Virtualization (NFV)
Software-Defined Networking (SDN)
Software-defined clouds
Service Function Chaining

ABSTRACT

The emergence of Network Function Virtualization (NFV) enabled decoupling network functionality from dedicated hardware and placing them upon generic computing resources. Moreover, the introduction of edge computing paradigm which utilized the resources at the network edges brings reduced end-to-end latency. With these technologies, Virtualized Network Functions (VNFs) can be placed in anywhere either in the central clouds to utilize more resources or in the network edges to reduce the end-to-end latency.

In this work, we propose a dynamic resource provisioning algorithm for VNFs to utilize both edge and cloud resources. Adapting to dynamically changing network volumes, the algorithm automatically allocates resources in both the edge and the cloud for VNFs. The algorithm considers the latency requirement of different applications in the service function chain, which allows the latency-sensitive applications to reduce the end-to-end network delay by utilizing edge resources over the cloud. We evaluate our algorithm in the simulation environment with large-scale web application workloads and compare with the state-of-the-art baseline algorithm. The result shows that the proposed algorithm reduces the end-to-end response time by processing 77.9% more packets in the edge nodes compared to the application non-aware algorithm.

1. Introduction

In the past decade, cloud computing has attracted huge attention from both industry and academia that enables utility-based computing resource provisioning with the advancement of virtualization technologies (Buyya et al., 2009). Physical resources, such as CPU cores, memory, storage, and networks, are virtualized and leased to cloud tenants who pay for only their usage instead of buying physical machines and paying for up-front installation fees. Application providers can rent only necessary amount of the resources from cloud providers in the initial deployment stage and easily scale up the application when more resources are necessary to support more users and data.

In addition to the advancement of cloud computing, a new paradigm of edge computing has emerged that utilizes resources at the edge of the network (Hu et al., 2015). In edge computing, applications and services are fully or partially served upon resources located on the edge of the network, instead of entirely serviced by centralized resources in cloud data centers. It reduces the end-to-end network delay by responding to the request from edge nodes instead of traversing the core network to reach the cloud data center. Also, it can reduce the network traffic between the edge and the central cloud by preprocessing and filtering the raw data generated by data source especially for the Internet-of-Things (IoT) and streaming applications. Although edge computing has brought many benefits, its limited resource capacity causes many challenges for efficient resource management. As the edge nodes have significantly limited resources compared to the central clouds, resource allocation and provisioning among edges and clouds are challenging to efficiently utilize both resources while increasing the benefits of edge computing.

Network Function Virtualization (NFV) (ETSI, 2018a) has been enabled from the advancement of virtualization technologies including the emergence of Software-Defined Networking (SDN) (Son and Buyya, 2018) and the increased hardware performance. Traditionally, network functions (e.g., firewalls, proxy servers, and intrusion detection systems) run on powerful and proprietary hardware dedicated for the particular network function. Network providers have to purchase the purpose-built device only for a network function and upgrade periodically as the network traffic grows up. Instead of using dedicated hardware, a Virtualized Network Function (VNF) utilizes common computing resources as

* Corresponding author.
E-mail addresses: json1@unimelb.edu.au (J. Son), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.jss.2019.02.030
0164-1212/© 2019 Elsevier Inc. All rights reserved.

exploited in cloud computing in NFV paradigm. A VNF runs upon generic computing resources to provide the same network functionality. Similar to Virtual Machines (VMs) in cloud computing, VNFs can be elastically scaled and migrated to different physical machines that enables dynamic and efficient resource provisioning based on the actual demand. Network providers running their own cloud data centers can utilize the cloud resources to place VNFs.

Although most research on resource allocation for VNFs focus on placing on centralized data centers (Herrera and Botero, 2016), utilizing edge resources for VNFs has been recently started and explored in the literature (Boubendir et al., 2016; Dominicini et al., 2017; Cziva and Pezaros, 2017). Placing VNFs in edge nodes carries out benefits for end users, application providers, and network operators. End users can get the response faster by placing VNFs in the edge nodes, as the network delay is reduced by avoiding additional network transmission from the edge to the central cloud for the network function. Reducing network latency obviously brings benefits for network providers and application providers as they can provide a better quality of service (QoS) to their respective customers.

However, the resource limitation on edge nodes brings challenges on resource management for VNFs. As we cannot place every VNF only onto edge nodes due to the limited capacity, the initial resource selection has to be carefully decided between edge and cloud resources for a VNF. For example, VNFs for latency-sensitive applications can be placed on edge nodes, whereas VNFs for less sensitive applications can be placed on cloud data centers. A recent work has explored the optimal placement for VNFs in edge and cloud computing (Cziva et al., 2018).

After the initial placement, it is also important to provision the resources dynamically adapting to the dynamic changes of the network loads. Network traffics passing through a VNF is dynamically changing depending on the number of connections and the volume of requests. Therefore, the provisioning method must be able to adapt to the change of the network traffic which can be significantly different from the initial placement. The method has to consider the limited resource size of the edge nodes and utilize the cloud resources when the required resources for a VNF exceeds the capacity in edge nodes.

In this paper, we propose a dynamic resource provisioning method for VNFs in the edge cloud environment. If no more resource is available in edge nodes, our algorithm creates a new VNF in the cloud data centers to distribute the network load. Furthermore, we consider the latency requirement of different applications utilizing the same VNF. Packets for latency-sensitive applications are forwarded to the original VNF in edge nodes up to the capacity limitation in order to maximize the benefit of edge computing. Packets for less sensitive applications divert to the new VNFs created in clouds to utilize abundant resources provided by the cloud data center.

The key **contributions** of this paper are:

- a dynamic resource provisioning methods for VNFs utilizing both edge and central cloud resources to deal with the resource overloading;
- a latency-aware VNF placement algorithm in edge clouds which selects edge resources over clouds for time-critical applications;
- an architectural design that manages both edge and cloud resources to provide minimum latency time for critical applications;
- a performance evaluation of the proposed algorithm through detailed simulation experiments in order to depict the effectiveness and improvements over the baseline algorithm.

The rest of the paper is organized as follows: Section 2 explains the background of edge computing and NFV, followed by exploring related works in Section 3. We propose the system architecture in Section 4 and the new algorithm in Section 5. Section 6 describes the experiment environment and performance results evaluating the proposed algorithm. Finally, Section 7 concludes the paper with the future directions.

2. Background

In this section, we explain the background of edge computing, Network Function Virtualization, and placing VNF in edge computing environment.

2.1. Edge computing

In edge computing environment, applications and service functions can be placed in edge nodes which can reduce the latency and the onward network traffics. The service latency can be reduced by placing the applications and service functions near to the end users. Instead of traversing to the central cloud resources over the high-latency WAN, edge nodes can instantly respond to the user request with minimum network latency. Since the service is placed next to the end user, network latency decreases. A similar concept has been widely adopted in the industry in contents delivery network (CDN).

Another benefit is to reduce the size of the network traffics. For example, in Big-data and IoT applications, edge nodes can perform filtering and data transformation tasks before transmission which can reduce the number of packets and their sizes. By filtering unnecessary data in edge nodes, the number of jobs sending onward to the central computing facility (i.e., cloud data center) can be reduced. Also, data transformation in edge nodes can compact the data size.

Although edge computing can improve the application end-to-end latency and network efficiency, it has limited resources compared to the central clouds. Thus, it is critical to optimize resource allocation and provisioning to maximize the benefit of edge computing. In this work, we consider edge nodes to place network functions to reduce the latency for time-critical applications.

2.2. Network Function Virtualization (NFV) and Service Function Chaining (SFC)

Network functions require high throughput and low latency in its processing, so that network providers had to exploit expensive dedicated hardware only to process network functions. However, the advancement of virtualization technologies and computing performance, network functions can be virtualized and served through commodity computing resources. Instead of buying high-cost purpose-built network equipment, network operators can process the same function in virtual machines running in generic cloud computing resources.

Although it is feasible to run Virtualized Network Functions on commodity hardware, resource management for VNF becomes critical in order to maximize resource utilization and minimize operational cost. For example, the resource amount for a VNF has to be decided to provide the required performance. Also, where to place a VNF becomes another issue that need to be addressed with a smart decision for the minimum network delay and network traverse. Even if the initial placement was optimal, VNF should be able to scale up and down based on the dynamic volume of the network traffic. For increasing network load, a VNF has to be able to scale up and/or duplicated to be able to process all the increased traffic on time without dropping packets.

In many cases, the series of network functions, which is also called service functions (SF), create a chain to serve the application. Service Function Chaining (SFC) is a chain of various network

Table 1
Summary of related works in VNF placement and provisioning.

Work	SF chaining	Dynamic provisioning	Domain	Parameters	Application consideration
Algorithm H (Xia et al., 2015)	✓	x	Single data center	Chain's connectivity	NA
Yang et al. (2016, 2018)	x	✓	Edges	Latency requirement	Mobile application
ABA (Bhamare et al., 2017)	✓	✓	Edges and clouds	Network traffic and affinity	NA
Cziva et al. (2018)	✓	✓	Edges and clouds	Latency	NA
Our approach	✓	✓	Edges and clouds	Real-time utilization and latency	Time-critical application

functions for certain traffics which performs various network functionalities (Pham et al., 2018). For example, an SFC can contain a sequence of NAT, Firewall, Intrusion Detection System, and Proxy, in which the packets for an application should go through every network function in the chain. As the network traffic has to go through the chained functions in an SFC, it is critical to consider the network affinity between the connected functions in SFC placement problem (Xia et al., 2015).

Utilizing NFV technology in edge and cloud environment has been explored in many studies (Boubendir et al., 2016; Dominicini et al., 2017; van Lingen et al., 2017; Cziva and Pezaros, 2017; Riggio et al., 2018). Placing a VNF in edge nodes can reduce the additional network delay brought by the network function. If all VNFs are placed in a central cloud, extra delays are expected for packets to traverse through the backbone network to reach the cloud data center before reaching to the application provider. VNF in edge nodes can minimize the extra latency.

However, computing resources at edge nodes are limited, and workloads are dynamic with little or no prior knowledge. If the number of network requests increases, and the VNF cannot handle them properly, it is necessary to provision cloud resources with more resources to serve the network function. In this work, we propose a novel algorithm to dynamically provision the edge and cloud resources to scale VNFs adapting to the dynamic workload. We use the network latency requirement of different applications to decide which resource to be exploited between edge and clouds for a specific type of applications.

3. Related work

A number of studies focused on VNF placement and resource management algorithm in the literature as depicted in Table 1. Xia et al. (2015) studied VNF placement algorithms for NFV chaining in a data center. Considering the connectivity between VNFs, the authors proposed an algorithm to assign VNFs within the same function chain onto the network pod in the topology to reduce the network traffic between the VNFs in a chain. The heuristic algorithm selects a pod within a data center to place a VNF, however, it has no consideration for the distributed multi-cloud or edge cloud environments.

Yang et al. (2016, 2018) proposed dynamic resource allocation framework with virtualized edge resources for mobile applications. The authors consider mobile applications that can dynamically move around multiple mobile edge-clouds and decides to increase or decrease the amount of resource for the application. The proposed framework also includes the re-optimization methods which periodically migrate and scale the existing functions to consolidate the resources and reduce the operational costs. Although the study explored the resource placement and dynamic provisioning problem thoroughly, the work has no consideration in Service Function Chaining and different application requirements sharing the same network functions. In contrast, our work focuses on provisioning VNFs consisting of a network function chain and tries to minimize the network latency for a time-critical application among various applications.

Optimal placement for VNFs in a multi-cloud environment has been studied by Bhamare et al. (2017) which considers the edge cloud environment. The authors presented the analytical model and a heuristic algorithm, named ABA (affinity-based allocation) for the service function chain placement problem in the multi-cloud environment. The proposed algorithm exploits the fraction of the network traffic for a VNF and their affinity in order to decide where to place the VNF. Evaluation results showed that their affinity-based heuristics can outperform other simple heuristics. This study focuses on the initial placement of VNFs in the multi-cloud environment, whereas our work is for provisioning resources dynamically to adapt to the ever-changing network traffic after the initial placement.

More recently, Cziva et al. (2018) presented a VNF placement algorithm at the network edge that considers network latency. The proposed work considers dynamically relocating end-users in multi-access edge computing environment. Given the network topology, the edge resources, and connection latency, the algorithm decides VNF to resource mapping: which VNF to be placed in which resources. The optimal solution for initial placement is presented with integer-linear programming model (ILP). In addition, a dynamic extension is proposed to decide the frequency of VNF migration and reallocation in case of user's movements and latency changes. Although the proposed method can result in the optimal solution, the cost and complexity of running the ILP algorithm can be significantly high in practice. Instead, we propose a heuristic algorithm for dynamic scaling and load-balancing for VNFs in the edge-cloud environment along with the latency-aware VNF forwarding algorithm in this paper.

Also, many platforms have been proposed for utilizing edge resources to serve VNFs (Mijumbi et al., 2016; Taleb et al., 2017). van Lingen et al. (2017) proposed a model-driven approach for bridging cloud and edge for NFV and fog computing. The model is based on ETSI's MANO architecture (ETSI, 2018b; Mijumbi et al., 2016) and includes fog computing as a part of the convergence. The platform can manage IoT services across the cloud and the edge. Riggio et al. (2018) also proposed a MANO based framework, named LightMANO, to deploy NFV in a distributed environment such as Multi-access Edge Computing environment. The proposed platform is implemented as a proof-of-concept on a small scale. The authors also explored the challenges of NFV deployment in multiple, distributed, and heterogeneous environment.

VirtPhy is another NFV orchestration architecture for edge data centers (Dominicini et al., 2017). The proposed architecture is capable of programming the orchestration strategy with the information of the network topology for geographically distributed small data centers in edge computing. The authors implemented a proof-of-concept system using OpenStack (OpenStack Foundation, 2017). Their architecture fits in ETSI MANO standard.

Cziva and Pezaros (2017) proposed a container-based NFV platform, named Glasgow Network Functions, to place and migrate a network function in the network edge nodes as a container. The platform can orchestrate the containerized VNFs to save core network utilization and reduce the network latency. The system is implemented upon various existing technologies, such as OpenDaylight SDN controller and OpenVSwitch upon Linux kernel. The authors also explored the VNF migration techniques with the specified timeline.

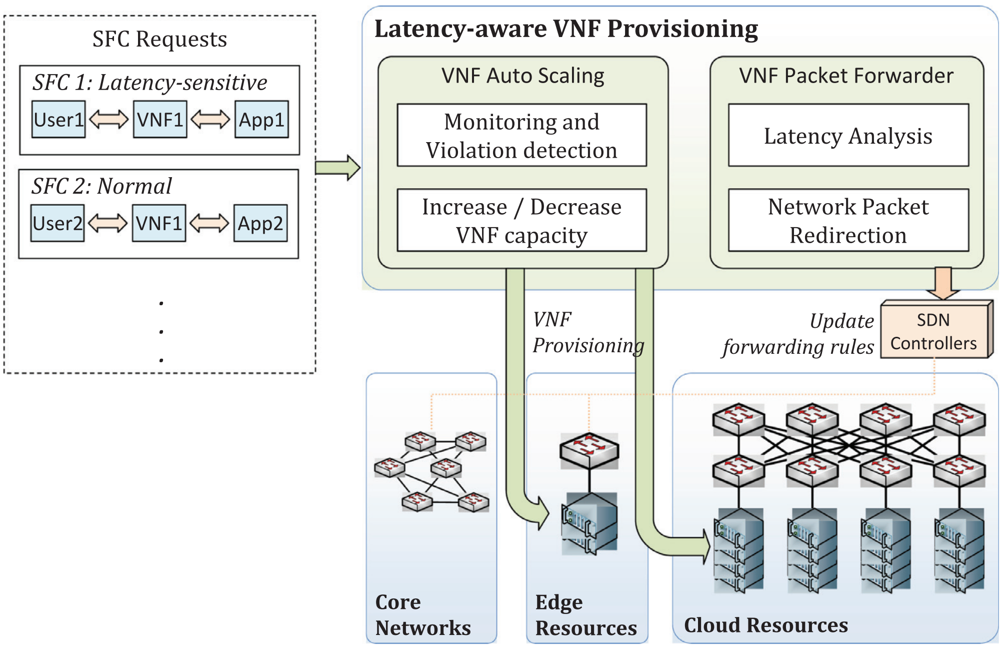


Fig. 1. VNF provisioning system architecture.

In this work, we propose a similar architecture for utilizing NFV in edge cloud environment. Our architecture enables that the edge resources can be utilized and orchestrated for NFV in addition to the central cloud. In addition to architecture, we focus on the development of new resource management algorithm which can utilize both edge and cloud resources.

4. NFV-enabled edge-cloud architecture

As we discussed in the previous section, various frameworks have been proposed in the literature for enabling NFV in edge-cloud environment (Boubendir et al., 2016; Dominicini et al., 2017; van Lingen et al., 2017; Cziva and Pezaros, 2017; Riggio et al., 2018). Building on them, we propose a system to dynamically scale VNFs in edge-clouds by duplication and forward network traffic to the duplicated VNFs with the latency information in the application requirement. Fig. 1 shows the proposed architecture. The system receives SFC requests consisting of the source, destination, and VNF chain of the application, which also includes the latency requirement of the different applications. For example, an application with a tighter delay requirement is classified as a latency-sensitive application. Thus, network packets for this application have to be processed with priority. We compare the latency requirement of different applications in order to figure out if the application is more delay-critical than others. If the delay requirements of an application is lower than the median of all applications sharing the same VNF, the application is regarded as time-critical for the VNF.

The system periodically monitors the current status of edge and cloud resources. If an SLA violation and an overloaded VNF is detected, it increases the size of VNF by duplicating it. If the edge has enough resource, it creates the duplicated VNF in the edge node, otherwise in clouds. Once the VNF duplication process is completed, the additional VNF can be used for load distribution through VNF Packet Forwarder. The forwarder updates forwarding rules in SDN switches across the core network, edge, and clouds. In order to decide which VNF to use, the system measures and analyzes the network latency from the source in the service chain to the alternative VNFs. The analyzed information is used to determine the alternative VNF in the forwarder module.

Fig. 2 shows an example scenario of the proposed system. Initially, two applications (App1 and App2) are served through the network managed by our system. Application servers are located on the internet whereas the end-users are in the edge site. Limited resources are available to run the VNF in the edge. The VNF in the edge site has to be passed through for both applications.

Once the network traffic going through the VNF increases, the capacity in the edge resources might not be enough to process all the increased network traffic. In this case, we have to utilize the cloud resource to create a duplicated VNF. Assuming the VNF is stateless, we can utilize either the original VNF or the duplicated VNF for same functionality regardless of the application. The proposed system determines which VNF to use by analyzing the latency from the end-users to each candidate VNF, as well as the application requirement. In this example, we assume that App1 is more time-critical than App2, and the original VNF in the edge site has less delay than the duplicated VNF in the cloud. With the analysis, the system set the forwarding rules in every site to forward the packets for App1 to the original VNF, and the packets for App2 to the duplicated VNF. The algorithm details are explained in the following section.

5. Latency-aware VNF provisioning for edge clouds

With the required latency information provided by the application providers, network operators can decide which resources to be exploited for the VNFs between edge and cloud. In principle, network packets for delay sensitive applications are sent through VNFs in edge nodes as far as the resources provided by edge nodes can handle the volume of the network traffic. Less sensitive packets can be distributed to the cloud data center to reduce the load of edge nodes. In this section, we describe our proposed algorithms in detail with pseudo codes and baseline algorithms including the state-of-the-art in the literature.

5.1. Latency-aware VNF provisioning algorithm

We propose a VNF provisioning algorithm in edge cloud computing environment which considers different latency requirements of applications. Although VNF placed in edge nodes can

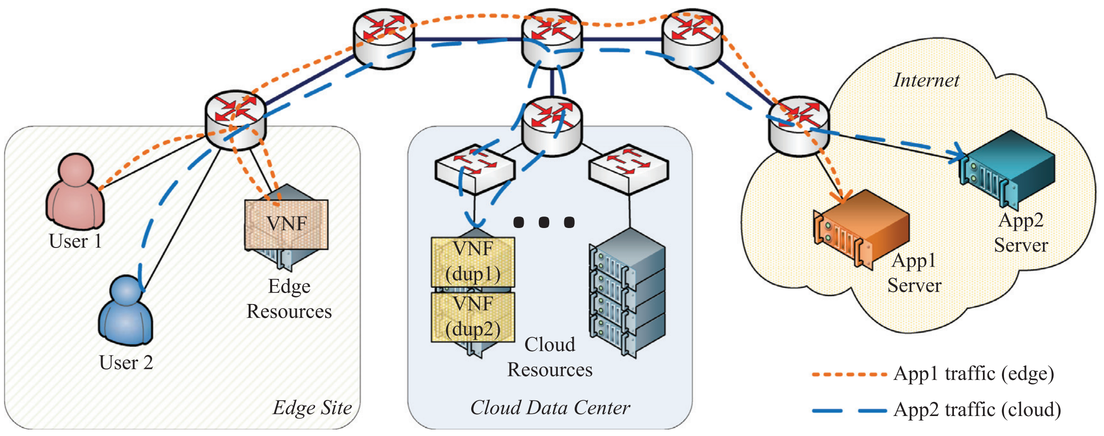


Fig. 2. VNF placement example in edge clouds.

reduce the network latency, computing resources are limited at edge nodes. On the other hand, the central cloud data centers can provide a large amount of resources for VNFs, but the network traffic must go through a number of extra hops to reach the cloud data center. Considering these trade-offs, we propose a latency-aware VNF provisioning algorithm to utilize edge resources for latency-sensitive applications while using clouds for less sensitive applications.

Our algorithm utilizes edge resources if the resources are enough to provide the amount of request. In the case of resource outage in the edge, the algorithm tries to divert some workloads to the cloud in order to distribute the load onto VNFs with enough resources. For latency-sensitive applications, we still utilize edge resources for meeting the required latency time. Less latency-sensitive requests are redirected to VNFs placed in the central cloud to utilize its adequate resources.

Algorithm 1 describes the overall process of the proposed algorithm. At first, the algorithm detects the overloaded VNFs that need more resources due to the dynamically increased amount of the network packets. The resource utilization of VNFs are constantly monitored and periodically detects the VNF overload, e.g. every 5 min. Once a VNF overload is detected, the algorithm duplicates the VNF for load-distribution in the same location if there are available resources. If the VNF located on the edge node is overloaded, for example, the algorithm tries to create another VNF in the edge. If the available resource in the edge is enough for the additional VNF, the edge node will run another VNF for the same network function, and the network packets are forwarded to either VNF regardless of the application's latency requirement. However, in the case if the resource is not enough in the edge, the new VNF will be placed in the cloud which increases network delay. In either case, the algorithm updates a latency map which will be used for forwarding the network packets.

Algorithm 1 VNF auto-scaling and provisioning for edge-clouds.

```
1:  Data: $VNF$: Currently running VNFs.
2:  for each VNF $v$ in $VNF$ do
3:      if $v$ is over-utilized then
4:          $loc \leftarrow$ Location of $v$ (edge or cloud);
5:          if $loc$.hasAvailableResource($v$) then
6:              $v' \leftarrow$ duplicate VNF $v$ in $loc$;
7:          else
8:              $v' \leftarrow$ duplicate VNF $v$ in cloud;
9:          end if
10:         Update latency map for $v$ to add $v'$;
11:     end if
12: end for
```

If the duplicated VNFs are placed in different locations, the VNF forwarder considers the application requirements to decide where to forward the network packets. When the duplicated VNF is placed in a different location, our algorithm creates a network latency map between the source of the packets and the VNFs in different locations to be used in the forwarder. It measures the network latency (e.g., ping time) from the source node of the network packet to the VNF in the edge and in the cloud. By preparing the latency map at the time of VNF duplication, the forwarder can use the latency information to decide a VNF to forward the network packet.

In order to keep the latency information between source nodes and the new VNF, we update the network latency map from the source node to different VNF alternatives. Please note that updating latency map is only performed right after the duplication, in order to reduce the overhead at the packet forwarding. As we prepare the latency map and sort them only at the time of VNF duplication, the VNF forwarder can simply select the preferred VNF among duplicated ones. Algorithm 2 presents the detailed pseudo-code for building a latency map. The algorithm has the original VNF and its duplicated alternative as input and measures the latency from the source node of all SFCs passing through the VNF. In detail, the algorithm starts to retrieve all SFCs that contain the input VNF. For each chain, we retrieve the previous latency map and get the source node of the chain which sends the packet to the VNF. Once we know the source node, we can measure the network latency from the source node to the newly duplicated VNF. The new VNF is inserted to the latency map along with the measured latency. At last, the map is sorted by the latency, from lower latency to higher latency, so that the forwarder can choose the front ones in order to reduce the latency.

Algorithm 2 Build latency map for a duplicated VNF.

```
1:  Input: $vnf$: Original VNF to be scaled.
2:  Input: $vnf'$: Newly created VNF duplicated from $vnf$.
3:  Output: $V_{all}(c)$: List of duplicated VNFs sorted by latency.
4:  $C \leftarrow$ All SFCs passing through $vnf$;
5:  for each SFC $c$ in $C$ do
6:      $V_{all}(c) \leftarrow$ Duplicated VNFs of $vnf$ for $c$;
7:      $src \leftarrow$ Source node in $c$ sending a packet to $vnf$;
8:      $l(src, vnf') \leftarrow$ Get latency from $src$ to $vnf'$;
9:      $V_{all}(c)$.insert($V'$, $l(src, vnf')$);
10:     sort ($V_{all}(c)$, key=$l(src, v)$ for $v \in V_{all}(c)$);
11: end for
```

Once duplicating VNFs and building a latency map are completed, newly arrived packets can exploit the new VNFs. VNF forwarder is in charge of distributing network packets whose destination is the VNF. If the VNF has no duplicate, it simply forwards all packets to the original VNF. If the VNF has been duplicated from the previous monitoring and auto-scaling process, the forwarder has to decide which VNF alternative to be used for a packet. When the network packet arrives at the forwarder, the forwarder decides which VNF to be selected by looking up the latency map and the application's latency requirements.

The proposed VNF selection algorithm to forward a network packet is presented in Algorithm 3. Latency-critical packets are forwarded to the VNFs with lower latency whereas packets for latency insensitive applications are forwarded to VNFs with higher latency value in the map. Note that the network load balancing is performed through the forwarder regardless of the application latency requirements. The proposed algorithm considers the various network latency requirements as well as the measured delay to VNFs in different locations in the process of the load balancing.

Algorithm 3 Latency-aware VNF selection algorithm.

```
1: Input: $c$: VNF chain to be forwarded.
2: Input: $vnf$: Original VNF for $c$ with duplicates.
3: Input: $C_t$: Count threshold for latency-sensitive traffic.
4: Output: $vnf'$: Selected VNF from alternatives of $vnf$.
5: $q \leftarrow$ Get count-based priority queue of duplicated VNFs for $vnf$;
6: if $q$ is empty then
7:     $V_{vnf} \leftarrow$ Get all duplicated VNFs of $vnf$ ;
8:     for each VNF $v$ in $V_vnf$ do
9:         $q$.insert($v$, $v_{count}$=0);
10:    end for
11: end if
12: $vnf' \leftarrow q$.poll();
13: if $c$ is latency sensitive then
14:     $V_{all}(c) \leftarrow$ Latency-sorted VNFs lists for $c$;
15:     for each $vnf''$ in $V_{all}(c)$ do
16:         if $|count(vnf'') - count(vnf')| < C_t$ then
17:             $vnf' \leftarrow vnf''$;
18:             break;
19:         end if
20:     end for
21: end if
22: $q$.insert($vnf'$, $vnf'_{count}$+1);
```

5.2. Baseline algorithm

In order to evaluate our algorithm, we implemented a baseline algorithm incorporated with the state-of-the-art work recently presented in Cziva et al. (2018). The authors utilize both edge and cloud resources to provision VNFs. In the case of VNF overloading in edge nodes, the algorithm migrates VNFs to the central cloud in order to utilize more resources. However, the work is in lack of consideration in application's latency requirements which can restrict the migration to cloud. Migrating all network traffics to clouds regardless of the application's latency requirement can lead to failures in responding within the boundary time for latency-sensitive applications. Our algorithm considers different latency requirements of the applications which makes the workload for latency-sensitive applications run in the edge nodes to keep their short latency, whereas the other workload to be redirected to cloud to utilize its adequate resources. The baseline is noted as Cloud+Edge algorithm in the following section for simplicity, which denotes that the algorithm utilizes both cloud and edge resources but without consideration of latency requirements.

6. Performance evaluation

We evaluated the proposed algorithm in a simulation environment to test its feasibility and effectiveness. For evaluation environment setup, we use CloudSimSDN (Son et al., 2015), a CloudSim (Calheiros et al., 2011) based simulation framework to support SDN and network functionalities. As the original CloudSimSDN does not support simulating multiple cloud data centers and network traffics, we extended and implemented several components to enable evaluating multi-cloud environment. The extended simulation software is able to simulate inter-cloud network transmissions, VNF creation and migration, and VNF forwarding policies. In this section, we present the experiment simulation environment, application configuration, and the evaluation results.

6.1. Environment, configuration, and workload

We created an edge site, a cloud data center, and the internet to simulate internet applications in the simulation environment as shown in Fig. 3. Application servers are located on the internet, which receives requests from end-users at the edge site. The edge site also has limited computing resources which can be utilized to place VNFs. A cloud data center is created with a large amount of resources so that VNFs can be running in the cloud if the edge site is lack of resources. In edge and cloud site, we put several network switches to build a local canonical tree network topology. Wide area network is set up to interconnect the edge, cloud, and the application servers on the internet. Network latency between hops is set to 1 ms for local area networks (within edge and cloud) and 100 ms for wide area networks to simulate the delay.

Upon the physical settings, two applications are prepared to represent latency-sensitive and normal applications. App 1 is a latency-sensitive application with tight delay requirement, whereas App 2 has a higher latency requirement. Each application consists of 12 servers placed on the internet to respond to the user requests. End-users are placed in the edge site and send application requests to the application servers. Network packets from an end-user to the application server passes through a chain of the network functions. We put a firewall for all packets from the end-user to the application server regardless of the application. The firewall is initially placed in the edge node. In short, a network packet is generated from the end-user in the edge site and then sent to the firewall in the same edge site. After processing in the firewall, it is sent to the application on the internet passing through the wide area network with higher latency. Please note that VNFs can be placed either in the edge node or in cloud resources. The response packets from the application server also pass through an Intrusion Detection System before arriving at the end-user. These VNFs are shared by two separate applications.

We generated two workloads to submit to each application. The workload is generated with a three-tier application model (Ersoz et al., 2007) with the 24-h Wikipedia traces available from *Page view statistics for Wikimedia projects*. Approximately 1.5 to 2 million requests are sent to each application server for 24 h with varied inter-arrival rate derived from the model.

6.2. Analysis of latency

We measured the response time of every request in the workload. Fig. 4a shows the average response time of all workloads regardless of the application with different VNF provisioning algorithms. The first algorithm (Cloud-only) indicates the result with only cloud resources exploited without using any edge resources.

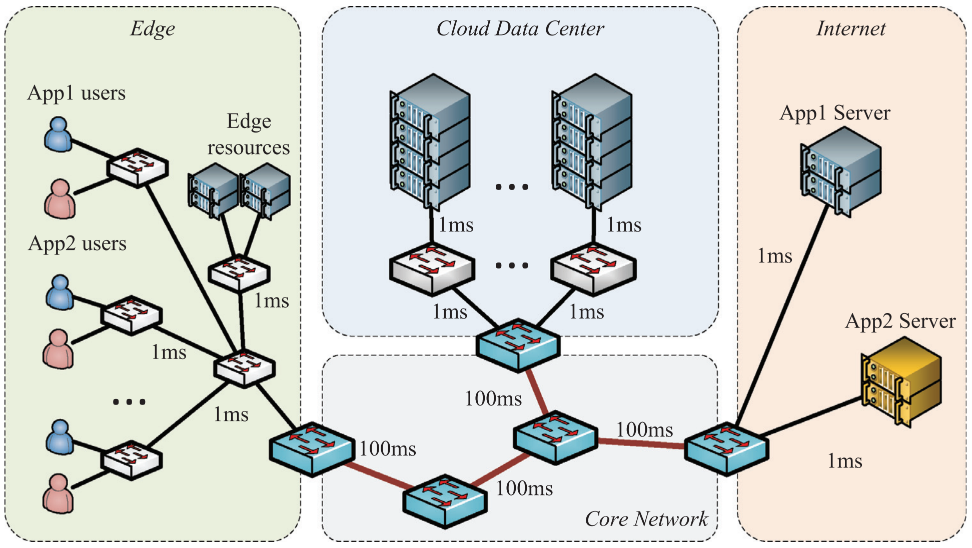
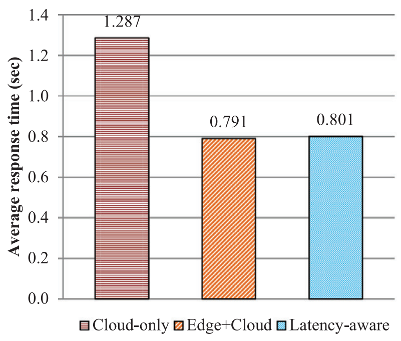
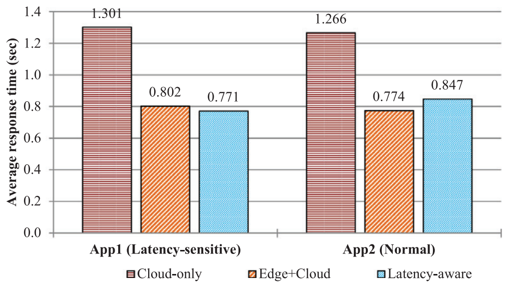


Fig. 3. Testbed configuration for simulation-based evaluation.



(a) Average response time for all applications.



(b) Average response time of each application.

Fig. 4. Average response time of workloads in different algorithms.

VNFs are created and provisioned only in the cloud resources, thus the average response time is significantly increased due to the additional delay that all the packets have to be transmitted to the central cloud. The next two results are with exploiting edge resources along with the cloud resources with the baseline algorithm (Edge+Cloud) and the proposed algorithm (Latency-aware). For both cases, the average response time is reduced to approximately 0.8 s compared to 1.287 s of the Cloud-only case. As the edge resources are utilized to process a part of the network traffic, those packets processed in the edge contribute to reducing the average response time. Overall response time for both applications is almost same for the baseline (Edge+Cloud) and the proposed latency-aware algorithm because the total number of packets forwarded to the edge resources are similar for load balancing in both algorithms.

When we measure the average response time for each application (see Fig. 4b) separately, the average delay is differentiated between applications. For the latency-sensitive application (App1), the average response time is reduced to 771 ms with the proposed algorithm. Compared to the baseline, it is 31 ms faster response for the time-critical application. Although the improvement percentage (3.9%) in this experiment scenario is relatively small, the algorithm has a potential for different scenarios on the different scale. Time-critical applications such as cyber surgery and road traffic management system could be benefited from the improvement if the proposed concept can be adopted to the application with refinement. However, the average response time of App2 is increased to 847 ms with our algorithm. As we explained, the proposed algorithm forwards more network packets from the critical application to the edge resources which help to reduce the latency. In contrast, a smaller number of packets can be processed in edge nodes for App2 for load-balancing purposes because the resources have been already taken by App1's packets.

6.3. *Analysis of packet proportion*

We also measured the proportion of the total network packets which are processed in either edge or cloud. Fig. 5 shows the percentage of the network packets processed in edge or cloud resources. When we placed VNFs only in clouds without utilizing any edge resources, obviously all packets are processed in clouds regardless of the application. When we utilize edge and cloud resources together (Edge+Cloud), where edge resources are utilized without knowledge of latency requirement, the similar portion of the packets of App1 and App2 are processed in edge resources. In detail, 43.7% of App1 packets and 43.9% of App2 packets are forwarded to the VNFs in edge nodes, whereas the rest of the packets are processed in the cloud.

In contrast, 78.1% of App1 packets are processed in edge resources with our latency-aware VNF provisioning algorithm. In

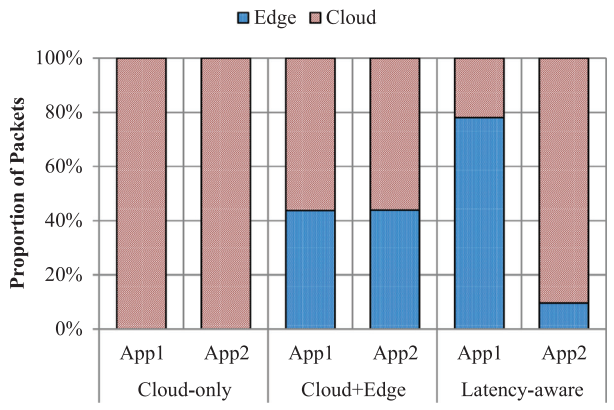


Fig. 5. Proportion of the network packets processed either in the edge or the cloud with different algorithms.

other words, 77.9% more packets are forwarded to edge resources compared to the baseline results. As more portion of packets is processed in the edge resources, the latency-sensitive application (App1) results in reduced delay on average for the end-users. On the other hand, only 10% of App2 packets are processed in the edge, and the other 90% is forwarded to the cloud. Therefore, the network delay for App2 would be increased as a trade-off which is acceptable and expected since App2 is a latency-insensitive application.

7. Conclusions and future work

Utilizing edge resources for Network Function Virtualization brings many advantages for end-users who needs a tight network delay requirement. Network operators can place VNFs in edge resources, instead of central clouds, in order to reduce the additional network delay. However, resource capacity in the edge is significantly limited compared to the cloud. Thus, it is critical to managing and provision resources efficiently for the distributed edge cloud environment.

In this paper, we explored dynamic resource provisioning methods for VNF in the distributed edge and cloud environment. The system architecture and the provisioning and forwarding mechanism are presented. We proposed a dynamic resource provisioning algorithm that utilizes both edge and cloud resources for dynamically increasing network demands. The algorithm also includes latency-aware packet forwarding method to utilize near-the-source resources for latency sensitive applications. The proposed algorithm is evaluated in the simulation environment with large-scale workloads derived from Wikipedia web application. Our results show that the proposed algorithm can put more workloads to edge resources for latency-sensitive applications which results in reducing average network delay for the application. We also showed that 77.9% more numbers of the network packets are forwarded to the edge for the time-sensitive application compared to the baseline.

Our algorithm can be extended and improved to include more parameters in the decision making. It takes the latency requirement submitted by application providers to decide which packets to be forwarded to the edge. In the process, we consider the half of tighter delay requirements as a critical application and the other half regards as a normal application. This can be improved by employing a linear approach instead of the binary decision. Also, in addition to the latency requirement, the algorithm can include other information of the application such as acceptable error rate, required bandwidth, and power source of different sites to decide which resource to be utilized.

Acknowledgments

This work is supported through a collaborative research agreement between the University of Melbourne and Huawei Technologies Co., Ltd (China) as part of the Huawei Innovation Research Program (HIRP). The work is carried out within the Melbourne CLOUDS Laboratory at the University of Melbourne. We thank Dr. Adel Nadjaran Toosi for the valuable feedback and discussion to improve the quality of the paper.

References

Bhamare, D., Samaka, M., Erbad, A., Jain, R., Gupta, L., Chan, H.A., 2017. Optimal virtual network function placement in multi-cloud service function chaining architecture. Comput. Commun. 102, 1–16. doi:10.1016/j.comcom.2017.02.011.

Boubendir, A., Bertin, E., Simoni, N., 2016. On-demand, dynamic and at-the-edge vnf deployment model application to web real-time communications. In: 2016 12th International Conference on Network and Service Management (CNSM), pp. 318–323. doi:10.1109/CNSM.2016.7818440.

Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I., 2009. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst. 25 (6), 599–616.

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. 41 (1), 23–50.

Cziva, R., Anagnostopoulos, C., Pezaros, D.P., 2018. Dynamic, latency-optimal vnf placement at the network edge. 2018 IEEE INFOCOM. IEEE.

Cziva, R., Pezaros, D.P., 2017. Container network functions: bringing nfv to the network edge. IEEE Commun. Mag. 55 (6), 24–31. doi:10.1109/MCOM.2017.1601039.

Dominicini, C.K., Vassoler, G.L., Meneses, L.F., Villaca, R.S., Ribeiro, M.R.N., Martinello, M., 2017. Virtphy: fully programmable nfv orchestration architecture for edge data centers. IEEE Trans. Netw. Serv. Manage. 14 (4), 817–830. doi:10.1109/TNSM.2017.2756062.

Ersoz, D., Yousif, M.S., Das, C.R., 2007. Characterizing network traffic in a cluster-based, multi-tier data center. In: Proceedings of the 27th International Conference on Distributed Computing Systems. IEEE. 59–59.

ETSI, 2018a. Network functions virtualisation, https://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano.

ETSI, 2018b. Open source NFV management and orchestration (MANO), https://www.etsi.org/technologies-clusters/technologies/nfv/open-source-mano.

Herrera, J.G., Botero, J.F., 2016. Resource allocation in nfv: a comprehensive survey. IEEE Trans. Netw. Serv. Manage. 13 (3), 518–532. doi:10.1109/TNSM.2016.2598420.

Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V., 2015. Mobile Edge Computing - A Key Technology Towards 5g, 11. ETSI, pp. 1–16. white paper.

van Lingen, F., Yannuzzi, M., Jain, A., Irons-Mclean, R., Lluch, O., Carrera, D., Perez, J.L., Gutierrez, A., Montero, D., Marti, J., Maso, R., Rodriguez, J.P., 2017. The unavoidable convergence of nfv, 5g, and fog: a model-driven approach to bridge cloud and edge. IEEE Commun. Mag. 55 (8), 28–35. doi:10.1109/MCOM.2017.1600907.

Mijumbi, R., Serrat, J., Gorricho, J., Latre, S., Charalambides, M., Lopez, D., 2016. Management and orchestration challenges in network functions virtualization. IEEE Commun. Mag. 54 (1), 98–105. doi:10.1109/MCOM.2016.7378433.

OpenStack Foundation, 2017. Open source software for creating private and public clouds., https://www.openstack.org/.

Pham, C., Tran, N.H., Ren, S., Saad, W., Hong, C.S., 2018. Traffic-aware and energy-efficient vnf placement for service chaining: joint sampling and matching approach. IEEE Trans. Serv. Comput.. 1–1. doi: 10.1109/TSC.2017.2671867.

Riggio, R., Khan, S.N., Subramanya, T., Yahia, I.G.B., Lopez, D., 2018. Lightmano: Converging nfv and sdn at the edges of the network. In: 2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9. doi:10.1109/NOMS.2018.8406266.

Son, J., Buyya, R., 2018. A taxonomy of software-defined networking (sdn)-enabled cloud computing. ACM Comput. Surv. 51 (3), 59:1–59:36.

Son, J., Dastjerdi, A.V., Calheiros, R.N., Ji, X., Yoon, Y., Buyya, R., 2015. CloudSimSDN: modeling and simulation of software-defined cloud data centers. In: Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 475–484. doi:10.1109/CCGrid.2015.87.

Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., Sabella, D., 2017. On multi-access edge computing: a survey of the emerging 5g network edge cloud architecture and orchestration. IEEE Commun. Surv. Tutorials 19 (3), 1657–1681. doi:10.1109/COMST.2017.2705720.

Xia, M., Shirazipour, M., Zhang, Y., Green, H., Takacs, A., 2015. Network function placement for nfv chaining in packet/optical datacenters. J. Lightwave Technol. 33 (8), 1565–1570. doi:10.1109/JLT.2015.2388585.

Yang, B., Chai, W.K., Pavlou, G., Katsaros, K.V., 2016. Seamless support of low latency mobile applications with nfv-enabled mobile edge-cloud. In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), pp. 136–141. doi:10.1109/CloudNet.2016.21.

Yang, B., Chai, W.K., Xu, Z., Katsaros, K.V., Pavlou, G., 2018. Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications. IEEE Trans. Netw. Serv. Manage. 15 (1), 475–488. doi:10.1109/TNSM.2018.2790081.