

Dynamic Equilibrium Capacity across Multiple Regions in NFV Environment

Hai Nguyen Dinh*, Sang-il Kim *, Hwa-Sung Kim*

*Department of Electronics and Communications Engineering, Kwangwoon University, Korea

nguyendinhhai11@gmail.com, rlatkd234@kw.ac.kr, hwkim@kw.ac.kr

Abstract— Recently, most emerging challenges associated with the Network Functions Virtualisation (NFV) are related to the automated, large and scalable deployment of virtualised network functions (VNFs), their availability as well as reliability. In this manner, VNF applications are vastly distributed to allow high accessibility to users all over the world. Therefore, a VNF application that has a distributed deployment over multiple datacenters located at the different regions, can perfectly fit needs and requirements of provider. In order to efficiently resolve the most emerging challenges to meet requirements of service provider (SP), efficient and optimal resource capacity management across multiple regions of NFV services is critical aspect. This paper proposed a dynamic equilibrium capacity method for automatically managing virtualized resources assigned to each project that is spread over different regions. This brings significant benefits to users to manage capacity more accurately and maximize the acceptance ratio of requests in a long run while minimizing the usage of resource efficiently, which leads to use a minimum deployment cost responding to user demands.

Keywords— NFV, PlaynetMANO, Dynamic Equilibrium, Capacity Management, Multiple Regions

I. INTRODUCTION

NFV represents a major shift in the telecommunications and networking industry. Many SPs are now keen to implement NFV to help them gain advantages. To reduce cost, enterprises are striving to consolidate their data-centers. This is not always the optimal choice for NFV. Many NFV applications require a real-time response with low latency. For instance, Virtual Evolved Packet Core (vEPC) or some of the high-performance VNFs such as Session Border Controller (SBC) and Advanced Media Software (AMS) utilized for multiple applications (VoLTE MRF and Interconnects, vIMS, etc.) are designed as distributed applications. For the heavy load applications (such as computer-aided design (CAD) modelling, video streaming, etc.), they are very bandwidth sensitive, often have multiple branches of the enterprise in different locations, and need to collaborate together. Therefore, thousands of edge clouds need to be arranged to place those kind of applications close to end-users for better user experience. In this context, the edge clouds serving for the enterprise are also essentially distributed to allow the VNFs to operate with a full dynamic elasticity that scales with traffic, optimizes resources utilization and provides low-latency services. Whereas, single region deployment will challenge in the capacity expansion that should be controllable,

modularized in NFV infrastructure (NFVI). Especially, NFV applications also need to be highly available and survive disasters. SPs need the flexibility to deploy network functions in a distributed infrastructure at the network core, metro area, and possibly even a premises of customer.

NFV infrastructure is built in fully distributed manner across multiple geographical regions to place workload close to the end user for better user experience, however, this kind of deployment model also introduces some cross-regional requirements: centralized management for distributed resources, centralized capacity management, centralized monitoring service, service function chaining, VNF placement, high availability and disaster recovery. How these requirements will be supported must be studied, analysed and well understood. Information provided in this paper aims to provide a method that would help to address the requirements in centralized management for distributed resources, centralized capacity management and centralized monitoring resources crossing regions. The European Tele-communications Standards Institute-NFV Industry Specification Group (ETSI-NFV ISG) [1][2] has defined framework for the management and orchestration (MANO) of all virtual resources in the NFV. This includes computing, networking, and storage. One of the most important core requirements in this architecture is resource capacity management. In the previous works, many proposals [3][6][7] focus on resource capacity management in cloud computing or NFV. However, most of proposals just focus on the optimal capacity management within single datacenter at a specific region.

In this paper, we concentrate on how to manage capacity of virtualized resources that are assigned to particular projects and distributed across different geographical regions. A proposed algorithm to dynamically manage and rebalance capacity of distributed resources across regions is introduced, it helps to provide operator with a joint view about resource utilization, total capacity per project and increase accuracy, as well as efficiency of capacity management. The objectives of our algorithm are to maximize the fitness values of the resource availability, which leads to maximum acceptance ratio of requests in a long run. In other words, failure rate of resource requests that occurs due to resource exhaustion will be minimized and resource utilization is optimized efficiently, that leads to use a minimum deployment cost to serve all user demands.

II. BACKGROUND

The following topics are major interest in the context of this paper: resilient multi-regions deployment of components integrated in PlaynetMANO platform, the regions concept as understood by NFV management system, management of distributed resources across different regions and dynamic rebalancing capacity crossing regions by monitoring resource usage. Therefore, in the following we provide necessary background information, explanations and relevant references for these points.

A. Resource Management Defined by ETSI-NFV ISG

The NFV-MANO can be referenced in order to better understand the critical NFV requirements for SPs as follows: [i] the ability to deal efficiently with a distributed NFV infrastructure [8], which is necessary for assuring low signal latencies and disaster resiliency. This infrastructure must, however, also be manageable as a single distributed cloud with global views, single unified view for operator, multi regions capacity management, policies driven VNF placements, optimized resource utilization, dynamic service functions chaining across regions. [ii] the ability to automate the lifecycle of complex carrier applications with many component functions that have different deployment, scalability and healing requirements.

B. Capacity Management Defined by The ETSI NFV ISG

The NFV orchestrator (NFVO) needs to know available capacity of an NFVI in order to make an informed decision on selecting a particular NFVI or Network Function Virtualization Infrastructure Point of Presence (NFVI-PoP). Such capacity information shall be in a coarser granularity than the respective Virtualized Infrastructure Manager (VIM), as VIM maintains capacity information of its NFVI in fine details. Capacity information may be required for the complete NFVI, or per region or partition or availability zone, or other granularities. Therefore, VIM requires to inform the NFVO about available capacity, resource utilization, capacity changes information regarding its NFVI at a pre-determined abstraction, either by a query-response, or in an event-based, or in a periodical way. A conventional property of cloud and NFV system is "capacity on demand". User from a given project is able to allocate new resources through user interface or API and to release them when the need ends. By supporting a large number of users and projects, pooling resources and maintaining some excess of capacities, SP presents the illusion of infinite capacity. However, in practice the resources are not infinite, and SP must institute measures to manage capacity so that resource exhaustion is minimized. This is done by imposing a cap (capacity) on the resources that a particular project may consume, and by managing the relationship between the available resources and the aggregate caps for all projects. When a project requires more resource than its assigned capacity, generally request is rejected or delays until sufficient capacity is available or require user to submit a request that needs human approval. When the request is approved, the capacity for the project is modified to reflect the new limit.

C. Region Concept

The concept of regions and groups of resources appear in different cloud management and orchestration systems. In this part, we describe how this concept is interpreted in these environments to clarify the terms of use and how our region concept in NFV system makes a connection to existing ones. As defined NFV ETSI ISG, region is mapped to one or more NFVI-PoPs where a VNF could be deployed. NFVI-PoP data centers are often constrained in compute and storage capacity. Since practically all NFVI-PoPs are foreseen to be distributed, inter-datacenter network capacity is also a constraint.

In Amazon Web Services (AWS), regions are actual geographical regions, e.g., Western Asia or North America, and are completely isolated from each other. In OpenStack, four different related concepts are defined: cells, regions, availability zones and host aggregates. While availability zones and host aggregates both serve the purpose of grouping of resources with or without attribute in common for isolation of components and assuring redundancy inside one region, cell and region are both used to distinguish between different sites of a cloud deployment: Cells share the same API endpoint from a client perspective, while regions have different API endpoints and do not cooperate. Recently regarding to discussion from various carriers who deploying OpenStack, cell mostly is appropriate to be used inside one region. Therefore, more relevant to our discussion is region.

In our PlaynetMANO architecture that will be described in the next section, regions are used by user or operator to provision resources. One region can act as one or more VIMs, PlaynetMANO can schedule and bind VNF of a given project inside one region. A project resources could be bound to multiple regional VIMs automatically.

D. Project Concept

A project represents a logical group of users to which resources are assigned. Resources which are assigned to project could be bound to different regions, not directly to users. A given project may be mapped to a particular business unit, a specific multi-tier application or even a single application tier. In these examples, project ultimately belongs to a single organization that can has multiple branches located at different regions or locations. Consequently, a user must be associated with an individual project. To prevent resource capacity from being exhausted without notification, cap is used in our architecture. Caps are operational limits for virtualized compute, storage and network resources. For example, the number of virtual CPU, virtual memory, virtual storage allowed for each project could be controlled so that resource capacities are optimized. Caps can be enforced directly at project level and indirectly at user level. As without sensible cap, a single project can use up all the available resources in resource pools (unlimited capacity). A user cap is defined in association with a specific project. The summation of all user caps under a same project should not exceed project cap. In our PlaynetMANO, NFVO and VNF Manager (VNFM) will support multi-projects (multi-tenancy) for VNF lifecycle operations. Project lifecycle

management operations are also supported: Create, read, update, and delete projects.

E. Resilient Multi-Regions Deployment of Components

In [2], different approaches are considered in NFV distributed deployment models supporting the placement of VNF. In this part, we introduce the resilient multi-regions deployment model which is adaptable to use in PlaynetMANO. For the reason that our platform is aimed to serve variety of complex NFV applications, especially carrier applications, which are deployed in distributed manner across regions to assure real-time response with low latency (by placing VNFs near users), high accessibility, reliability, as well as disaster recovery and redundancy capabilities to users, regardless of geographical location. This architecture describes the case where the NFVO is the only functional block to manage the virtualized resources needed for the lifecycle management of the VNFs.

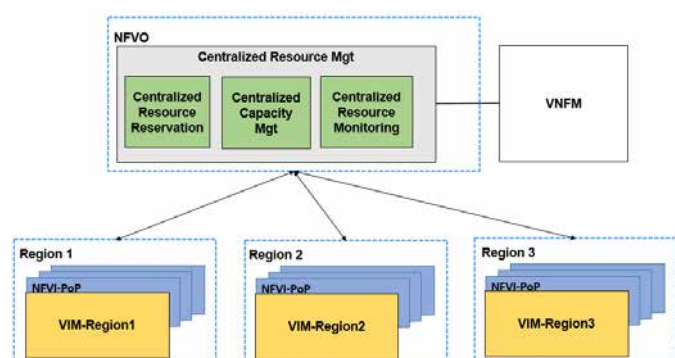


Figure 1. Resilient NFVO multi-regions deployment

The NFVO provides an overall view of the resources present in multiple regions to which it provides access and hides the interfaces of the VIMs present below it. The regions and caps that are present in the different regional VIMs are visible at the NFVO interfaces. In NFVO, global resource management (RM) is one of crucial roles of orchestration. This requires abstracted NFVI resource capacity information in order to take decisions on VNF placement and other operations related to the virtual resources. VIM is required to inform the NFVO of NFVI resource state information for this purpose. In resilient NFV deployment model, project is spreading over different regions as illustrated in Figure 1. Each region will be deployed with independent VIM OpenStack [5]. Using shared keystone [4] is often recommended. Additionally, a common carrier approach is to have one or multiple NFVI-PoPs in a region with limited shared services, this increases the overall region availability.

III. PLAYNETMANO AND SYSTEM MODEL

PlaynetMANO that is an ICT and ITP integrated project as described [9] introduces an innovative concept: Network Functions as-a-Service (NFaaS) by designing and implementing an integrated architecture, including the following components:

- **Playnet Launcher:** It includes a set of services to facilitate the involvement of multiple stakeholders through

PlaynetMANO infrastructure: A PlaynetStore where the third party developer VNFs are stored along with their metadata, a friendly graphical user interface (GUI) for the network service composition, management and monitoring as depicted in Figure 2.



Figure 2. PlaynetMANO UI/UX

- **PlaynetMANO:** PlaynetMANO is based on ETSI NFV-MANO specifications. It is not completely aligned yet with the interface specifications (IFA), as PlaynetMANO brings up some additional elements which are out of scope of ETSI NFV as expressed in Figure 3. It provides a Http-server module that interfacing with Playnet Launcher through Restful API, an NFVO building block that communicating with VIMs via a VIM connector, a generic VNF, components for Element Management (EM) and EM agent (EMA), supports OpenStack VIM and especially, improving VNFs performance significantly by leveraging Docker as hypervisor at virtualization layer along with KVM hypervisor. This allows PlaynetMANO to flexibly deploy VNFs either as container or Virtual Machine (VM).

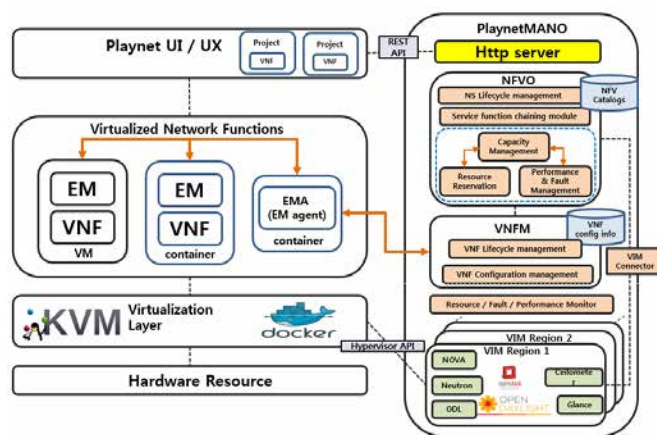


Figure 3. PlaynetMANO platform, module and interfaces

Our work mainly focuses on RM module in NFVO component. It consists of the following sub-modules:

- **Capacity Management:** It provides centralized capacity management service for project that is spread over different regions at VIMs.
- **Resource Reservation:** This enables PlaynetMANO to organize the access to resources based on time slicing. User

in a project is allowed to create, update, delete and query their own reservations. Time-scheduled reservation is used to provide resource booking for network services or VNFs in our PlaynetMANO architecture. Hence, user can schedule the most cost-effective actions to guarantee customer services better while improving the efficiency of virtual resource utilization into future.

- **Performance and Fault Management:** A module provides centralized collection, persistence and fault handling of infrastructure related information available from an infrastructure virtualization, management layer and VNFs built with open source components (Zabbix, etc.).

IV. DYNAMIC MANAGEMENT OF DISTRIBUTED RESOURCE CAPACITY IN NFVO

Multi-projects deployment to avoid RM race conditions. The same user may have multiple VNFs in individual project across multiple regions, it is essential to centralize all the resources consumed by these VNFs. Currently, there is no centralized management for distributed resources across multiple regions in NFV. A user should be able to manage all their virtual resources from one centralized management interface in NFVO, to have a summarized view of the total resource capacity, the live utilization and availability of their virtual resources. Consequently, these will facilitate NFV infrastructure to support VNF placement in selection of a region that best fits to resource requirements of VNF. In the following part, we introduce proposed algorithm which is to provide operator/user a global resource utilization and total capacity per project view, improves accuracy and efficiency in managing capacity of distributed resources across different regions. As result from our experiment, failure rate that occurs because either resource is exhausted or resource pool is not well prepared for user demand have been minimized. Additionally, resource utilization and deployment cost are optimized compared to the greedy approach using Best-Fit decreasing method (BFD).

A. Centralized Monitoring Resource Utilization in NFVO

For the real-time metrics, there is a need to have a central point to monitor the resource utilization and reservation (vCPU, volume, memory, port etc.). Users may have multiple VNFs in different geographical regions, hence, it is required to centralize all the consumed resources and logs generated by these VNFs. In our PlaynetMANO, Zabbix that is the ultimate enterprise-level software designed for real-time monitoring of millions of metrics collected from tens of thousands of servers, virtual machines and network devices has been used for monitoring solution.

B. Dynamic Rebalancing Capacity across Multi-Regions

Limitation of cap is used to set operational limits to prevent system capacity from being exhausted without notification. For the capacity limit itself, a centralized management is required. Otherwise, we have to do the capacity limit per region base, losing the advantage to have a centralized RM service. Furthermore, with [3] it's difficult to make limits in all regions keep consistency, and control distributed usage not exceed the limit in a diffusive and cumulative manners. Generally, all caps

Algorithms: Dynamic Rebalancing Capacity Management

Input: A list of projects (number of projects: $N \geq 1$)

Result: Balance capacity of each project binding to all regions over the time.

Create a project task list $P_j = [P_1, P_2, \dots, P_j, \dots, P_n]$;

For $j \leftarrow 1$ **to** n **do**

```

     $R_{ij} = [R_{1j}, R_{2j}, \dots, R_{ij}, \dots, R_{mj}]$ ;  $\leftarrow$  Finding the list of all regions
    binding to specific project  $j^{th}$ ;
    GetSum ( $R_{ij}$ ):  $SUM_{ijk} \leftarrow$  Calculating utilization of resource type  $k$ 
    at each region  $i^{th}$ ;
     $SUM_{jk} \leftarrow$  Calculating summation of resource (type  $k$ ) usage of
    project  $j^{th}$ ;
     $GPC_{jk} \leftarrow$  Getting and setting global cap of resource (type  $k$ )
    assigned to project  $j^{th}$  to its all regions;
     $GRC_{jk} \leftarrow$  Calculating global remaining cap of resource (type  $k$ )
    assigned to project  $j^{th}$ ;
    Create a region task list;
    For  $i \leftarrow 1$  to  $m$  do
         $PRC_{ijk} \leftarrow$  Computing new equilibrium cap of resource type  $k$ 
        that is assigned to region  $i^{th}$  in project  $j^{th}$ ;
        Updating new equilibrium cap for project  $j^{th}$  in region  $i^{th}$ ;
    End
End

```

are set for each region separately and different type of resources (compute, storage, network, etc.) have different caps to be set. Cap that is assigned to a specific project is fragmented, distributed to different regions and currently has no rebalancing capability crossing multiple regions. Therefore, in this part we solve this problem by proposing an algorithm that automatically adjusts limitation of cap based on current resource usage collected from all regions where a project is bound to, as well as giving operator or user a global view about capacity per project across multiple regions. It is important to note that only the operational limits imposed by the caps will be aligned, consumption of caps by users will not be reflected between regions. By leveraging VIM OpenStack APIs and NFVO southbound API, the integration and communication among region VIMs and NFVO are much easier.

We assume that in our NFV architecture there are n projects, m regions where a project could be spread over. Each project and relevant regions have k type of virtualized resources (such as virtual CPU, virtual memory, virtual storage, virtual network, etc...). Let $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$ for all $i \in I$ and $j \in J$. Constrains of total demanding resources in each of project could not exceed cap which is assigned to project and will be automatically rebalanced at all related regions. Resource reservation is also a key service to launch the success of RM in NFV as mentioned in previous section. In our works, reserved resource (RE) is also taken into account. A total resource utilization SUM_{jk} (resource type k) of project j^{th} that are collected from relevant regions can be described as:

$$SUM_{ijk} = (RA_{ijk} + RE_{ijk}) \quad (1)$$

$$SUM_{jk} = \sum_{j=1}^n (\sum_{i=1}^m SUM_{ijk}) \quad (2)$$

Where RA_{ijk} is allocated resources of project j^{th} in region i^{th} and RE_{ijk} is reserved resources of project j^{th} in region i^{th} . As reserved resource is scheduled and held in a given period of time with specific start time and end time. Therefore, when starting time t_s arrives, that kicks off resource allocation from NFVO to VIMs, then RE_{ijk} would be transferred to RA_{ijk} starting from the time t_s . At the end, resources will be de-allocated when end time t_e is triggered.

TABLE 1. NOTATIONS

Symbol	Description
GPC_{jk}	Global cap of resource type k that is assigned to project j^{th}
GRC_{jk}	Global remaining cap of resource type k that is assigned to project j^{th}
RA_{ijk}	Allocated resource type k of project j^{th} from region i^{th}
RE_{ijk}	Reserved resource type k of project j^{th} from region i^{th}
SUM_{ijk}	Project j^{th} usage summation of resource type k at region i^{th}
SUM_{jk}	Project j^{th} usage summation of resource type k
PRC_{ijk}	New equilibrium cap of resource type k that is assigned to project j^{th} in region i^{th}

Based on these institutions, our solution consists of the following major steps: we first retrieve project usages from each of regions, then in the second step we perform the computation according to project usages from all relevant regions that includes the following sub-steps: obtaining the global capacity GPC_{jk} of resource type k that is initially assigned to project j , next we calculate global remaining capacity GRC_{jk} and consequently, computing new equilibrium capacity PRC_{ijk} of resource type k for corresponding project at each region. Finally, we rebalance capacity limits spreading over different regions through VIM-connector exposed by the VIMs and the NFVO in our PlaynetMANO architecture.

Step 1. Retrieving total project resource utilization: The goal of this step is to obtain total project resource utilization SUM_{jk} from all relative region branches. To calculate SUM_{jk} , firstly we need to finding the list of regions that project j^{th} is bound to and then computes resource utilization SUM_{ijk} (1) at each of region (i^{th}) by invoking $GetSum(R_{ij})$ function, where input parameter R_{ij} is region (i^{th}) Id. As a result, SUM_{jk} is the sum of SUM_{ijk} as formulation (2).

Step 2. Computing equilibrium capacity limits: In this step, we calculate the equilibrium capacity limits for project in each of region based on total (SUM_{jk}) and individual resource utilizations (SUM_{ijk}) retrieved in first step. In NFVO, The global capacity limits of resource type k (GPC_{jk}) that is assigned to project j^{th} is used to initiate each capacity limits of its own relevant projects j^{th} in all region branches equally. Then we determine the global remaining capacity (resource type k) of project j^{th} (GRC_{jk}) by comparing GPC_{jk} and SUM_{jk} as:

$$GRC_{jk} = \sum_{j=1}^n (GPC_{jk} - SUM_{jk}) \quad (3)$$

The goal of above sub-steps is to determine new equilibrium capacity limit PRC_{ijk} of project j^{th} in each of region branch by the sum of the global remaining capacity (resource type k) of project j^{th} (GRC_{jk}) to actual resource utilization SUM_{ijk} in each of corresponding region branch (that project j^{th} is bound to) to cover the already allocated and reserved resource in that region, expressing as:

$$PRC_{ijk} = \sum_{j=1}^n (GRC_{jk} + \sum_{i=1}^m (SUM_{ijk})) \quad (4)$$

Finally, we use PRC_{ijk} to automatically rebalance capacity for corresponding project j^{th} in region i^{th} . New equilibrium capacity limit PRC_{ijk} is set to project j^{th} in region branch i^{th} . The value of PRC variables in different region branches are different depending on actual resource usage in that region. This allows users from project j^{th} in region i^{th} can consume additional amount of resources up to (5) based on total global capacity limits (GPC_{jk}) stored in NFVO. Therefore, the fitness value of resource availability is maximized as formulated in objective function (5)

$$\text{Max} (PRC_{ijk} - SUM_{ijk}) \quad (5)$$

The reason is that there are SUM_{ijk} resources that are already allocated and reserved in the region i^{th} and $(GPC_{jk} - PRC_{ijk})$ is amount of resource in use in other regions (where project is also bound to). Similarly, the limits in region $(i+1)^{th}$, region $(i+2)^{th}$ and so on should be set in the same manner.

In other words, the total resource available in project as expressed in (5) that can be consumed by users from any regions is maximized in a best effort manner. Comparing with diffusive capacity limited in each of local region approach which enforces a global capacity by ensuring that the sum of all local capacities does not exceed the total capacity for a particular resource and guarantees that the local resource usage of an application process does not exceed the local capacity, our approach enables flexibility and readiness for resource pools to be well prepared for future resource demands from users, even with demand that requires to consume a huge of resources. Therefore, capacity management is optimized and the chances of a resource request failure is minimized. On the other hand, number of required region nodes is minimized that leads to use a minimum deployment cost responding to user requests.

V. EVALUATION

Simulation environment: We consider a scenario that system has different regions. There are dozens of projects which are spread over different regions. In this experiment, we assume that regions are connected to each other through a high speed link (low latency) and communicate with NFVO in our PlaynetMANO architecture through restful API.

Virtualized resources: In this evaluation, we assume that capacity are operational limits for virtualized compute, storage and networking resources (number of VNFs, vCPUs, vmemory,

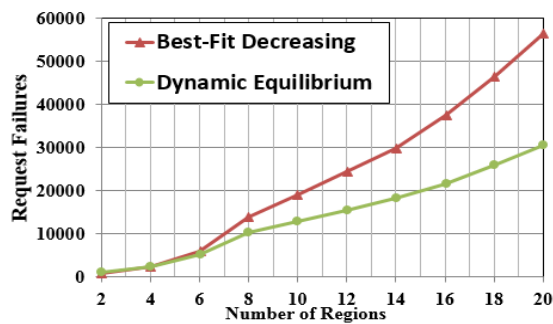


Figure 4. Number of request failures

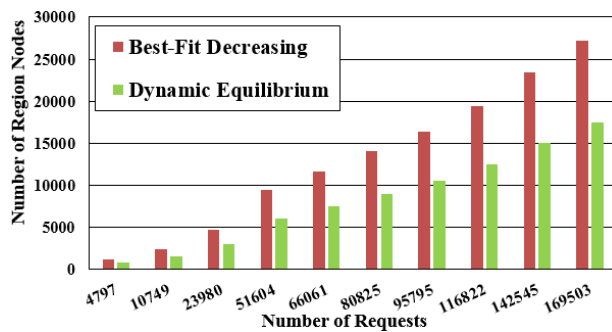


Figure 5. Number of used region nodes

snapshots, volumes, gigabytes, vport, vrouter etc.). We apply hot, warm and cold thresholds in sequence of 80%, 70%, 20% which are used to prevent resource capacity in project from underutilization or being exhausted without notification. Resource utilization in a specific project does not exceed more than 80% of the total resource available on that project. In other words, user in a specific project cannot consume up all available resources from resource pools. When a project requires more resource than its assigned capacity, in our work we assume that request is rejected.

To evaluate our method, we implemented a request generator to randomly generate up to 170000 user requests based on amount of available resource capacity assigned to a specific project. In this context, we assume that requests are randomly assigned to different regions and load-balancing controller in NFVO will be used to balance operational requests among separate regions. Through the evaluation, we compared our method with BFD method, which is widely studied in the literature for solving NP-Hard bin packing problem. In the BFD method, we firstly initialize local capacity for each of local region with equal amount of resources computed from global granted project capacity. For instance, if global granted capacity of project A is maximum 180 vCPUs, local capacity in each of region (total three regions) is 60 vCPUs. Then requests are sorted in the order of decreasing size, and best-fit procedures are applied for the requests in turn, respectively [10]. From experiment, our method is observed to be scalable (beyond 6 region nodes as depicted in Figure 4) while heuristic BFD method has limited applicability, especially to the case with large number of region nodes. Furthermore, dynamic rebalancing capacity approach flexibly manages and adjusts resource capacity against resource usage monitored over

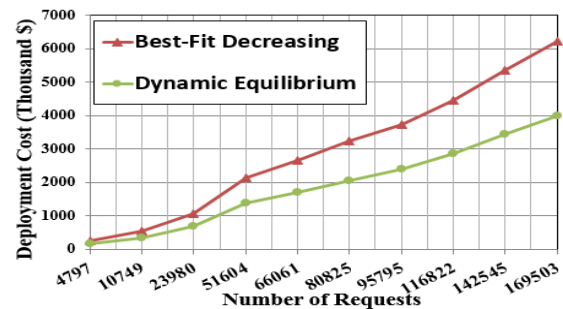


Figure 6. Deployment cost

periods of time, this enables resilience and readiness for resource pools to be well prepared for future resource demands, which leads to maximum acceptance ratio of requests. Therefore, capacity management is optimized and the chances of a resource request failure is minimized around 34% less than BFD as described in Figure 4, Figure 5 illustrates proposed approach reduces around 35% of required region nodes to host all user requests in comparison with BFD approach. Consequently, in terms of the total cost our method performs better than the BFD as stated in Figure 6.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented such a dynamic equilibrium capacity management method in NFV environment. In our method, the total available resource in project that users can request from any regions is maximized in a best effort manner, this helps to minimize request failure rate. On other hand, number of required region nodes is minimized as well that leads to use a minimum deployment cost to serve all user requests in comparison with heuristic greedy approach using best-fit decreasing method. The evaluation and analysis show that our method is adaptable and can be applied to capacity management of multiple regions NFV environment. However, when the number of projects and regions are large, the increment of concurrency requests and usage queries might require a considerable latency for query, computation or the latency imposed on a data packet travelling through network. Therefore, our future work will attempt to find more efficient way to reduce latency and computation time. Network capacity management among different regions (such as bandwidth, throughput, etc.) need to be studied further in our next steps. Alternatively, we intentionally plan to make our framework as centralized point that has ability to interface to other commercial VIMs such as Microsoft Azure, Amazon WS, and VMware to provide access to robust commercial high availability services and enabling users to choose resources from multiple providers.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MSIP (Ministry of Science, ICT and Future Planning)/IITP (Institute for Information & Communications Technology Promotion). [12221-14-1001, Next Generation Network Computing Platform Testbed].

REFERENCES

- [1] [ETSI-GS NFV-IFA005] "Network Functions Virtualization (NFV) Management and Orchestration", Proc. of European Telecommunications Standards Institute NFV Industry Specification group, April 2016
- [2] [ETSI-NFV-IFA009] "Report on Architectural Options, ETSI GS NFV-IFA 009 V1.1.1", Proc. of European Telecommunications Standards Institute NFV Industry Specification group, July 2016.
- [3] Johannes Behlaliz, Tobias Distler and Rudiger Kapitza, "DQMP: A Decentralized Protocol to Enforce Global Quotas in Cloud Environment", Proc. of the 14th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'12), vol. 7596, pp 217-231, Springer, 2012.
- [4] OpenStack Keystone. [Online]. Available: <https://wiki.OpenStack.org/wiki/Keystone> (visited on Jan. 1, 2017).
- [5] OpenStack Services Architecture. [Online]. Available: <http://docs.openstack.org/arch-design/multi-site-architecture.html>, (visited on Jan, 2017).
- [6] R.Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R.Boutaba, "Network function virtualization: State-of-the-art and research challenges", Proc. of IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.
- [7] R. Mijumbi, J. Serrat, J. l. Gorricho, S. Latre, M. Charalambides, "Management and orchestration challenges in network functions virtualization", Proc. of IEEE Communications Magazine, vol. 54, no. 1, pp. 98–105, January 2016.
- [8] Multi-domain Network Virtualization, [Online]. Available: <https://tools.ietf.org/pdf/draft-bernardos-nfvrg-multidomai-n-01>, Proc. of IETF Working Group, October 2016.
- [9] Sunghyun Yoon, Taeheum Na, Hosun Yoon, Ho-Yong Ryu, "Design and implementation of virtual network function development toolkit", Proc. of IEEE International Conference on Information and Communication Technology Convergence (ICTC), December 2016.
- [10] H. Iima, T. Yakawa "A New Design of Genetic Algorithm for Bin Packing", Proc. of IEEE International conference of Evolutionary Computation, 2003. CEC '03. The 2003 Congress on, May 2003.



Hai Nguyen Dinh is currently enrolled in MS. program at the Dept. of Electronics and Communication Engineering Kwangwoon University, Seoul, Korea. His research interests intelligent resource management in NFV/SDN-enabled network



Sang-il Kim is currently enrolled in Ph.D. program at the Dept. of Electronics and communications Engineering, Kwangwoon University, Seoul, Korea. His research interests Semantic web, context reasoning



Hwa-sung Kim received Ph.D degree at the Dept. of Computer Science, Lehigh University, Bethlehem, PA 18015 USA. He is now a professor at the Dept. of Electronics and communications Engineering, Kwangwoon University, Seoul, Korea. His research interests include mobile network protocol, mobile web computing, embedded software