Enhanced Real Time Content Delivery using vCPE and NFV Service Chaining

Pouya Yasrebi
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, ON, Canada
pouya.yasrebi@mail.utoronto.ca

Hadi Bannazadeh
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, ON, Canada
hadi.bannazadeh@utoronto.ca

Alberto Leon-Garcia
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, ON, Canada
alberto.leongarcia@utoronto.ca

***Abstract*—Real-time content delivery (RTCD) systems have become a prominent aspect of telecommunications as evidence by popularity of news-casting, real-time event subscription / publication and live media streaming. Unlike conventional content delivery systems, RTCDs need to deliver processed information to users in real time. This may require the network to handle some of the processing closer to the users to efficiently use the bandwidth consumed by the applications. The combination of Network Function Virtualization (NFV) and service chaining is a promising solution to address this challenge.**

Our work applies a service chaining algorithm to place NFV modules of an RTCD application in a Software Defined Infrastructure (SDI), where virtualized Customer Premise Edges (vCPEs), possessing scarce resources, are employed. We suggest containers to efficiently pack VNFs into vCPEs. Our objective is to maximize the total number of chains that can be serviced in the RTCD application. To optimally chain the NFV modules, a heuristic algorithm is proposed and evaluated.

Using simulations, we show that our algorithm with the help of vCPEs can support higher number of users while providing high-level service quality.

I. Introduction

Real-time content delivery (RTCD) systems support applications and new business models of sectors such as finance, security, and telecommunication. In security, face recognition algorithms determine identities and report flagged criminals to police departments. In telecommunication, the live content of sports events is streamed to video subscribers with low delay. RTCD systems are required to avoid content delivery delays for different geographical locations, minimize system inaccessibility in dynamic applications, and provide applications to a higher number of users.

To avoid excessive content delivery delays and enable buffering, computing and networking resources need to be provisioned to provide the considerable processing and significant bandwidths capacities required by RTCD networks. These resources have become an essential part of content upload, distribution, and streaming in settings where live content needs to be processed rapidly to meet customer needs for superior quality and lower delay.

Availability of dynamic RTCD applications can be improved by converged management of heterogeneous network and compute resources via a software-defined infrastructure (SDI). To cope with dynamic nature of RTCD applications, SDI efficiently handles configuration modifications for the application by rerouting network traffic, scaling computational resources or even fully migrating computational resources [1]. SDI also employs a unified interface that facilitates deployment and management of RTCD technologies such as monitoring and live broadcasting [2]. The architecture of an SDI implementation shall be used to enhance RTCD in this work. The Smart Application on Virtual Infrastructure (SAVI) testbed is a fully deployed SDI resource management system (RMS). SAVI aims to support low-latency and high data volume applications based on a multi-tier cloud including Smart Edges. The notion of multi-tier cloud on SAVI extends to sensor networks with a third tier that contains virtualized Customer Premises Edges (vCPEs). vCPE's definition in SAVI is different from virtual

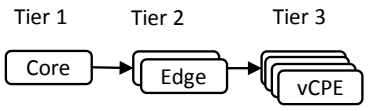

Fig. 1: Abstract multi-tier cloud

customer premises equipment, running fixed set of network functions. vCPEs in SAVI are referred to container hosting platforms located at the extreme Edge. vCPEs do not have enough resources to handle computation for high volumes of traffics, so a cloud-based solution with either Core or Smart Edge nodes is required to satisfy end-to-end requirements for an application.

Another recent technological breakthrough, which can revolutionize RTCD, is Network Function Virtualization (NFV) that aims to make a shift in the implementation of network functions [3]. NFV, a flexible networking architecture virtualizing network nodes, allows modular development and quick deployment. Each network module individually deploys a virtualized network function (VNF). VNFs require both computing and networking resources when deployed on the cloud. A network application might demand a set of these inter-connected VNFs to work together in a complex manner.

978-3-901882-85-2

The logic and process of connecting these VNFs together are called VNF service chaining [4]. There are two phases in VNF service chaining: placement and chaining [5]. VNF placement is concerned with the number of VNFs needed for an application and their mapping to physical servers. The VNF placement objective can vary from lowering deployment cost to maximizing traffic requests, and maximizing clustering of VNFs. Chaining is concerned with routing traffic through VNFs to minimize VNF chain traffic congestion, total processing, and propagation delay, and preserving Service Level Agreement.

Our work aims to create a scalable algorithm that processes submitted requests to the VNF chaining platform [6] of an RTCD application running on SAVI. The objective of this algorithm is to place and connect the VNFs in a way to maximize the capacity of submitted requests. The results of this algorithm note the importance of having vCPEs in increasing service coverage of the RTCD application.

The rest of paper is dedicated to studying the effect of vCPEs and Service chaining on an RTCD application. Section II discusses an RTCD application on SAVI and considers basic cases with and without vCPE to demonstrate the necessity of vCPEs on SDI. Section III describes the service chaining and provides a mathematical formulation for reaching to the optimal solution of service chaining capacity maximization problem. To make service chaining (an NP-hard problem) tractable, a heuristic solution has been proposed and evaluated in Section IV. Section V discusses related work and section VI concludes the paper.

II. System Description

Kaleidoscope [2] is an application that involves the gathering and distribution of video in live events to and from massive groups of users, and as such it is an ideal use case for an RTCD application on the SAVI testbed. Kaleidoscope can leverage dynamic cloud resource allocation and network configuration in real time to achieve efficient resource management and better service performance. Our work can be used for resource allocation and service chaining in Kaleidoscope.

We consider a stadium using Kaleidoscope (seen in Fig. 2). The stadium is equipped with high-quality cameras placed in different angles of the stadium. These cameras are connected to Wi-Fi access points and send video feeds to a streamer server. In this work, we solely focus on content distribution from the streamer-server to the fans in the stadium.

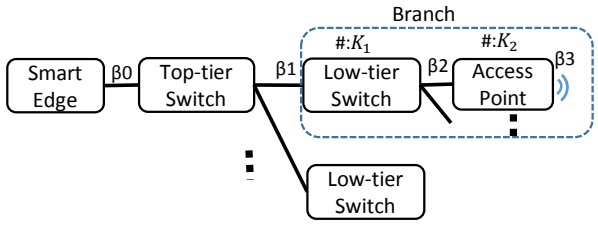

Fig. 2: Stadium physical topology considered in basic cases

In the setting under consideration, all users should be able to stream content captured by the cameras in the field at all times. In addition, a playback feature has to be available for a percentage of these users.

To enable stream and playback features, a set of traffic requests should be submitted to the VNF chaining platform of the SAVI testbed. These traffic requests contain service chain information for content distribution originated from a streamer-server. Each service chain traffic request includes deep packet inspection (DPI), HTTP-streamer, cache, and the streamer-server. Aside from the streamer-server, the locations of all VNFs are unknown. The streamer server is located on the Smart Edge as a single shared instance. Therefore, most of the requests will be directed from the vCPE nodes to the Smart Edge.

In our case an ordering for placement of all VNFs is necessary. Each user request is in a form of a strand, a small set of connected chains that form a graph. The VNF strands are as shown in Fig.3 . In the stream case, the DPI has to be the first module that a request from a user logically visits. This is to prevent harms that a request can cause on the SDI. It is important to note that DPI is only used on the uplink (having 5% of the downlink traffic) to avoid any contention. Following the DPI, a request should face an HTTP-streamer module. In this configuration, a separate HTTP-streamer VNF (for stream and replay) and Cache VNF (for replay) is needed for each request. These VNFs might change the nature and size of the bandwidth. DPIs forward data that is passed to them without changing the bandwidth of the flows.

Local caches reduce bandwidth consumption of playback feature. This local caching and computation illustrates the benefits of using vCPEs by saving on total bandwidth and allowing more users to stream. In order to more efficiently

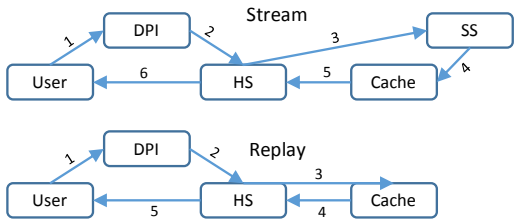

Fig. 3: Flow strands of stream and replay requests

use scarce resources on vCPE, we propose the use of the containers instead of VMs for deploying VNFs. In other words, VNFs will be deployed on containers that are running on vCPEs. [7].

Here we define multiple basic cases and find the maximum number of users (streaming and replaying with $b = 5Mbps$) that can be supported by the SDI. As seen in Fig.2, $\beta 0$ is the bandwidth between the Smart Edge and the top-tier switch, $\beta 1$ is the bandwidth between the top-tier switch and each low-tier switch, $\beta 2$ is the bandwidth between low tier switches and each of their connected access points, and $\beta 3$ is the bandwidth of wireless link connecting each access point to its user group.

K_1 is the number of low-tier switches and K_2 is the number of the access points per low-tier switch according to the figure.

Lets consider few numbers for a set of parameters to ease understanding the first few cases: $\beta 0 = 10$GigE, $\beta 1 = 1$GigE, $\beta 2 = 600$Mbps, $\beta 3 = 200$Mbps, $K_1 = 300$, $K_2 = 5$.

A. collocated chain model

First we assume all VNF modules of a strand have been collocated on the same compute module. The strands have been transformed to chains by ignoring the up-link due to its smaller bandwidth, and video has been multi-casted from streamer server to all possible locations for the HTTP-streamers (seen in Fig.4).

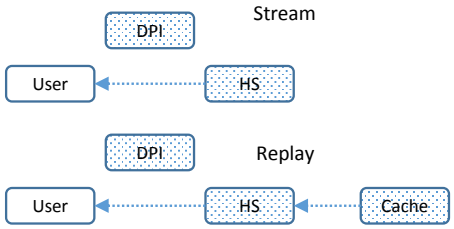

Fig. 4: simplified chains of stream and replay requests. Cache modules are considering processing power per user and do not have a local copy per user.

1) Smart Edge compute only: In this case, the bottleneck is the link between Smart Edge and the top-tier switch with bandwidth $\beta 0$. The SE can support up to $\beta 0/b$ users (2000 fans in our example).

2) vCPE enabled access point only: Despite the previous case, all VNFs have to be placed on the APs. Each user needs b bandwidth per stream or replay from the wireless bandwidths. Hence $(\beta 3/b)$ users can be supported by an AP accounting for $K_1 K_2 (\beta 3/b)$ in total(i.e.72000 fans in our example).

3) limited vCPEs only: Let's consider a case that only K_2' (i.e. 1) out of K2 APs are equipped with vCPEs. Therefore these vCPEs could be shared by their neighbor APs. The multicast used to distribute V (i.e.$V = 10$) camera feeds only consumes a bandwidth Vb from the link connecting low-tier switch to the vCPEs. $\beta 2 - Vb$ is available to service neighbor access points. Therefore number of users supported by the SDI (here ≈ 42000 fans) is limited to the capacity of wireless links (WIFI_{cap})and capacity of outgoing links from the APs with vCPEs ($\text{vCPE}_{\text{link-cap}}$)

$$\begin{aligned} &\text{minimum} && \text{WIFI}_{\text{cap}}, \text{vCPE}_{\text{link-cap}} \\ &\text{WIFI}_{\text{cap}} && K_1 K_2 (\beta 3/b) \\ &\text{vCPE}_{\text{link-cap}} && K_1 K_2' ((\beta 2 - Vb)/b + \beta 3/b) \end{aligned}$$

4) Smart Edge and limited vCPEs: In this case the bandwidth left from multicast on the main link shall be used to service the users. This modifies the previous formula to: (43980 users)

$$\begin{aligned} &\text{minimum} && \text{WIFI}_{\text{cap}}, \text{vCPE}_{\text{link-cap}} + \text{SE}_{\text{link-cap}} \\ &\text{SE}_{\text{link-cap}} && ((\beta 0 - Vb)/b) \end{aligned}$$

5) Processing limit on the vCPEs: Let's assume that collocated VNFs from a stream request chains require $res1$ amount of vCPU and each vCPE has vCPU capacity of RES1. The problem becomes

$$\begin{aligned} &\text{minimum} && \text{WIFI}_{\text{cap}}, \text{vCPE}_{\text{cap}} + \text{SE}_{\text{link-cap}} \\ &\text{vCPE}_{\text{cap}} && \min(\text{vCPE}_{\text{link-cap}}, \text{vCPE}_{\text{res-cap}}) \end{aligned}$$

where vCPE_{cap} is the number of users supported by resource and link capacity of vCPEs (i.e. RES1/res1 is the users supported by resources). To increase the vCPE_{cap}, it seems reasonable to distribute VNFs of the strands instead of collocating them.

B. Service Chaining on constrained heterogeneous resources

We have realized by now that reducing compute on the vCPEs of the AP can lead to more number of users. To efficiently use compute on the APs the location of DPIs and HTTP-streamers, as the main compute hungry VNFs, are of great importance. So far the effect of adding vCPEs on number of users was observed. The vCPEs and Switches with processing are becoming very common in today's networking research. It was notable that vCPEs do not need to be installed on each and every AP to provide a reasonable service to the users.

In this case, we are assuming APs and lower-tier switches are equipped with heterogeneous resources. Computing the maximum number of users that can be supported by the infrastructure may not be as straightforward as it was in previous cases. To explore the effect of service chaining on the maximization of the number of users, a mathematical formulation is developed in the next section.

III. Problem Formulation

Connectivity of servers(compute modules) S with direct physical links L is represented by a directional graph $G(S, L)$ in which server S_i is connected by link L_{ij} with effective bandwidth (chosen below 70% of actual rate to avoid queuing delay) of $\beta_{ij}(Gbps)$ and propagation delay of $\delta_{ij}(ms)$ to server S_j. Delay on the links (header processing delay) is comparably smaller than computational delays (delays involved in using RAM). This makes the total delay the sum of all processing delays and a constant number. The constant delay eliminates the need to discuss the notion of delay bound in our problem formulation. Each server has a set of resources $R = \{r_1, r_2, ...\}$ i.e: CPU (GHz), RAM (GB), Disk (GB), Disk Read Speed (Mbps) and a capacity associated with each resource C_s^r. Since abundant storage is available, the storage capacity is not considered as a bottleneck and is removed from the calculations.

In our use-case, a group of requests T is submitted to the RTCD application. Each request t contains an ingress point u_t, an order of VNF chain ζ^t, a vector of bandwidths β^t that could change in each VNF process. Every VNF $m \in \zeta^t$ has a set of required resources ρ_m^r that need to be fulfilled with the SDI. Also a vector of virtual links Γ^t is associated with each request. A hypothetical physical link from a server to itself,

modeling inter-server traffic forwarding, has been considered for collocated VNFs on servers.

To be fair between users, our goal is to equally maximize number of users that can stream among different branches. A branch is a sub-tree of the SDI tree (as seen in Fig. 2 with a low-tier switch as its root. Our topology has K_1 branches (equal to the number of lower-tier switches in the SDI). By defining $T_{\text{stream},i}$ as the number of users that can stream from an AP of branch i, our goal becomes maximizing the minimum of the $T_{\text{stream},i}$s among different branches. This problem is equivalent to maximizing number of chains per AP when all access points have similar number of chains:

maximize	T_{stream}
s.t	$T_{\text{stream}} = \alpha T_{\text{replay}}$
	$T_{\text{stream}} = T_{\text{stream},1} = ... = T_{\text{stream},k}$
	$T_{\text{replay}} = T_{\text{replay},1} = ... = T_{\text{replay},k}$
α	Replay percentage
constraints	bandwidth and compute limitations

The maximum number of chains depends on the allocation of VNFs onto the physical servers. To formulate this allocation, a parameter x^t_{ms} is needed to indicate whether VNF m from traffic t is assigned to server s or not. On the other hand, physical links that are used to connect VNFs together have limited capacity. To determine physical link l usage for virtual link γ, another parameter $y^t_{\gamma l}$ is necessary.

To ensure that traffic requests T are assigned and connected as submitted to the chaining service, additional constraints need to be applied to the optimization problem as described next.

Var.	Description
x^t_{ms}	VNF m from traffic t on server s placement decision variable
$y^t_{\gamma l}$	virtual link γ from traffic t on physical link l chaining decision variable

To guarantee that all VNFs for the stadium are placed on the network, each VNF has to be allocated to one and only one server:

$$\forall t \in T, m \in \zeta^t$$
$$\sum_s x^t_{ms} = 1$$

Allocated containers to a server should require less resources than what is available on that server:

$$\forall s \in S, r \in R$$
$$\sum_t \sum_{m \in \zeta^t} x^t_{ms} * \rho^r_m \leq C^r_S$$

Each virtual link connects a VNF to another. To avoid redundant traverses, only a single direction of a physical link should be associated to a virtual link ($\bar{l}$ is the reverse direction of physical link l).

$$\forall l \in L, t \in T, \gamma \in \Gamma^t$$
$$y^t_{\gamma l} + y^t_{\gamma \bar{l}} \leq 1$$

Virtual links that are mapped to a physical link should require less bandwidth than what that link provides:

$$\forall l \in L$$
$$\sum_{t \in T} \sum_{\gamma \in \Gamma^t} y^t_{\gamma l} * \beta^t \leq \beta_l$$

The following constraint assures that at least a physical link is used for each virtual link:

$$\forall t \in T, \gamma \in \Gamma^t$$
$$\sum_{l \in L} y^t_{\gamma l} \geq 1$$

Next one needs to assure that every link of the traffic requests is fully mapped to the physical network. This condition is sufficient only if two consecutive VNFs on a chain do not collocate. We also require full path connectivity by equivalent input/outputs for flows between nodes. The reason for separating sums in the below expression is to emphasize the inbound and outbound traffic. The source and destination of each traffic are considered in above constraint.

$$\forall t \in T, \gamma \in \Gamma^t, \{(i,j) | \gamma = (i,j)\},$$
$$\{(s,d) | (s,d) = l \in L, s \neq d\}$$
$$\sum_{d \in S} y^t_{(i,j)(s,d)} - \sum_{d \in S} y^t_{(i,j)(d,s)} = x^t_{is} - x^t_{js}$$

To consider inter-server links, the possibility of incorrect collocations are eliminated as follows:

$$\forall t \in T, \gamma \in \Gamma^t, \{(i,j) | \gamma = (i,j)\},$$
$$\{(s,d) | (s,d) = l \in L, s = d\}$$
$$y^t_{(i,j)(s,d)} \leq x^t_{is}$$
$$y^t_{(i,j)(s,d)} \leq x^t_{js}$$

IV. Heuristics

Due to the binary nature of variables and integer format of constraints, the mixed binary linear problem above is an NP-Hard problem (proof in the appendix). Since each VNF can be placed on n servers, the solution space grows exponentially with respect to the number of servers in the SDI service chaining problem. Due to the tree topology of the graph, the paths between two VNFs in a chain can be uniquely determined. This tree feature keeps the complexity at the node selection level. n^T possibilities for the ILP problem makes finding the optimal solution quite time-consuming and possibly unfeasible.

Based on the problem formulation we believed that considering a possibility of all servers for each VNF, and the possibility of all physical links for each virtual-link were the main reasons for increasing the size of the problem. It was possible to benefit from the tree nature of the SDI to control the size of the problem. Our goal was to abstract global branches as simple resource pools (black-box) for chains in a local branch. In this approach, each chain will use its own branch if possible before borrowing resources from rest of the branches.

We propose a two-step heuristic process for solving the problem: Move DPIs to the Smart Edge to efficiently utilize vCPEs, and next find a right balance between inner and outer chain capacities for each branch to maximize the number of users per AP.

The DPI is on the upstream with minimal bandwidth passing through it. To avoid waste of resource on the vCPEs, it seems legitimate to offload the DPIs to the Smart EDGE. Placing DPIs on the vCPEs could drastically decrease the number of users benefiting the vCPEs. The placement of the Http-streamers and the caches is still a question.

Now the balance between inner and outer chains per branch must be found. Inner chains are the chains located in a branch to support the users located in the same branch. With this definition, outer chains become the chains in a branch supporting users from other branches. A trade-off between inner and outer chains is apparent. Users in a branch can have their chains use resources from the local branch, global (other) branches, or even Smart Edge. A branch can either support global users using local resources or local users using global resources but not both at the same time, since a branch either has excess or lack of resources. The inner and outer chain view to the problem is valid since having HTTP-streamer and caches if placed in different branches, consumes a lot of networking resources in the SDI.

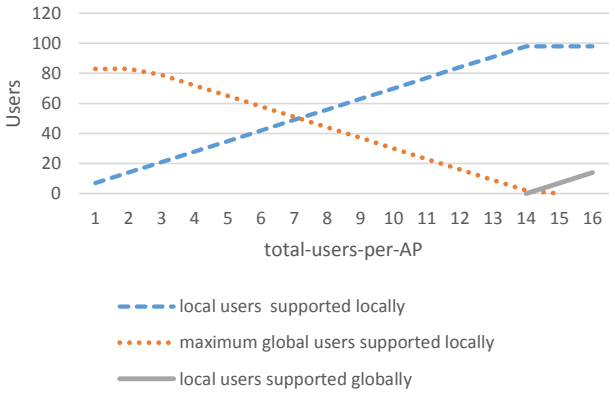

Fig. 5: lending chains to global branches

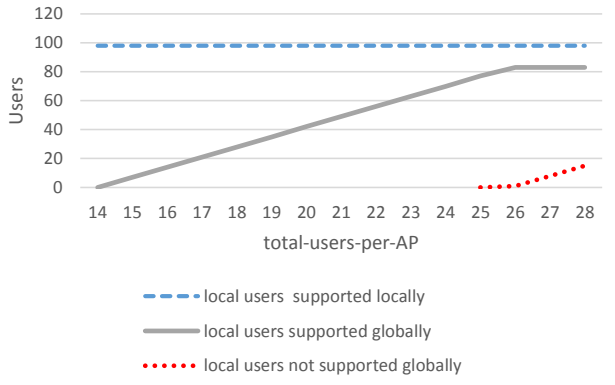

Fig. 6: borrowing chains from global branches

TABLE I: Number of users supported by Optimal (in total) and by Heuristic (per branch) are compared with the RAM required to solve the problems

	total users **in stadium**	RAM (GB)
Optimal	5	.4
	30	9.4
	200	?
	total users **per Branch**	RAM (GB)
Heuristic	5	.2
	30	.5
	200	9.3

To obtain a heuristic maximum number of the users, a similar set of figures as 5 and 6 has to be generated for each branch. These figures demonstrate the relationship between a number of users served per access point in a branch with respect to the number of local users supported locally, global users supported locally, local users supported globally, and local users that cannot be supported.

A VNF service chaining in SDI is valid where a number of total lent chains (possible global users supported locally) is higher than total borrowed chains (local users supported globally). This can be easily determined by combining all branch figures. Note that Smart Edge itself has a separate figure that only supports global users with local resources.

$$\max_{\text{total users per AP}} \sum_{i \in \text{Branches,Smart Edge}} \text{GL}_i$$

$$\text{GL}_i \leq \text{LG}_i$$

where

GL_i global users supported locally$_i$

LG_i local users supported globally$_i$

Table I compares RAM required to support optimal and heuristic algorithms for three use cases. It is completely notable that the optimal algorithm can only handle almost up-to 30 service chain in one shot, despite 200 users (200 $*$ number of branches in Total) for the heuristic solution. 40 users per AP in the heuristic scenario is enough for our problem.

The Heuristic solution can be applied to more general cases when each branch has an arbitrary topology with random resources. This allows the heuristic solution to be truly scalable (even for a stadium with 60000 users with 300 branches), while optimal solution falls short due to its exponential RAM Usage (seen in table I).

V. Related Work

[8] by Luizelli et al. divides the service chaining problem into three steps of placement, assignment, and chaining. This work solely considers reducing the number of VNFs on the infrastructure and it does not address service chaining's bandwidth consumption. [9] by Moens et al. considers a more general hybrid infrastructure with legacy hardware and cloud resources. It simply offloads work from legacy to the cloud as the capacity of the legacy is reached. [10] by Mehraghdam et al. considers traffic requests containing non-ordered VNFs.

To deal with the exponential growth of possibilities for non-ordered chains, this paper suggests a heuristic to select one of the possibilities. Maximizing remaining bandwidth for chain placement is one of the prominent problems that it considers. [11] by Huang et al. takes the same objective and proposes inter and intra-chain congestion control as the cause of running out of remaining bandwidth. Their service chain selection algorithm maximizes the remaining bandwidth. Finally, [12] by Bari et al. defines an ILP for the cost parameters in service chaining and takes an exhaustive approach in finding the minimum cost of service chaining in their network.

VI. Conclusion and Future work

In this paper, an algorithm to optimally map a set of VNF chains to the physical network of SDI for an RTCD application was designed. The heuristic reduces problem size to multiple local problems, resulting in a very efficient and scalable solution. The proposed scalable heuristic can support more arbitrary configurations than what we examined in this paper. In the future, we will consider placement of vCPEs from different flavors on the stadium.

VII. Appendix

Following is another formulation of the problem. By showing that the new formulation is an NP-Hard problem, we conclude that our original problem is also NP-Hard. We intend to utilize the underlying SDI to maximally contain Kaleidoscope VNF-graphs. The total number of VNF-graphs can be translated to maximization of the total number of users served by APs. Assuming ns_i,nr_i are the total number of users that can stream and replay in user group i, the objective would become:

$$\max \quad (ns + nr)$$

$$ns = ns_i \ , \ nr = nr_i$$

Lets consider a user group i with our specific VNF-graph shown in Fig.3. ns_{ij} is the number of streams from ns_i that have their DPIs located on node j. This would consume a bandwidth correlated with ns_{ij} from node i to node j, and a set of resources consumed on node j.

In a similar way ns_{ijk} is the number of streamers from ns_{ij} with their Http-Streamers on node k. An additional term nr_{ijkh} is used for the replay's cache component

$$ns_i = \sum_i (ns_{ij}) \ , \ ns_{ij} = \sum_k (ns_{ijk})$$

$$nr_i = \sum_i (nr_{ij}) \ , \ nr_{ij} = \sum_k (nr_{ijk}) \ , \ nr_{ijk} = \sum_h n_{ijkh}$$

$$(\alpha - \epsilon) ns_i \leq nr_i \leq (\alpha + \epsilon) ns_i$$

Due to the tree form of the topology there is a unique way to connect a node to another. W_{ij} represents paths used to connect node i to node j. Now we can construct formulation of bandwidth usage in our case.

Let's consider βs_i to be the bandwidth used for stream i, Bs_{UD} the bandwidth from user to the DPI,Bs_{DH} the bandwidth from DPI to the Http-Streamer, Bs_{HU} the bandwidth from Http-Streamer to the User-groups. βs_i can be formulated as follows:

$$\beta s^i = \sum_j ns_{ij} W_{ij} B_{UD} + \sum_j \sum_k ns_{ijk} (W_{jk} B_{DH} + W_{ki} B_{HU})$$

$$\beta r^i = \sum_j nr_{ij} W_{ij} B_{UD} + \sum_j \sum_k nr_{ijk} (W_{jk} B_{DH} + W_{ki} B_{HU}) + \sum_j \sum_k \sum_h nr_{ijkh} W_{hk} B_{CH}$$

$$Cs^i_q = \rho s_D (ns_{iq}) + \sum_j \rho s_H (ns_{ijq})$$

$$Cr^i_q = \rho r_D (nr_{iq}) + \sum_j \rho r_H (nr_{ijq}) + \sum_j \sum_k \rho r_C (nr_{ijkq})$$

With physical resource constraint of :

$$\text{Phys res.} \quad \sum_i Cs^i_q + Cr^i_q \leq C_q$$

$$\text{Netw res.} \quad \sum_i \beta s^i + \beta r^i \leq B$$

As the above problem is ILP (therefore NP-Hard) and equivalent with our original problem, the original problem is NP-Hard as well.

References

[1] T. Lin, B. Park, H. Bannazadeh, and A. Leon-Garcia, "Savi testbed architecture and federation," in *1st EAI International Conference on Future access enablers of ubiquitous and intelligent infrastructures (Fabulous)*, September 2015.

[2] Q. Zhang, S. Q. Zhang, J. Lin, H. Bannazadeh, and A. Leon-Garcia, "Kaleidoscope: Real-time content delivery in software defined infrastructures," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 686–692, May 2015.

[3] ETSI, "Network function virtualization - introductory white paper," October 2012.

[4] M. Chiosi and et.al, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, 2012.

[5] P. Quinn and T. Nadeau, "Service function chaining problem statement," *draft-ietf-sfc-problem-statement-07 (work in progress)*, 2014.

[6] P. Yasrebi, S. Bembey, H. Bannazadeh, and A. Leon-Garcia, "Virtual network function service chaining on savi sdi," in *1st EAI International Conference on Future access enablers of ubiquitous and intelligent infrastructures (Fabulous)*, September 2015.

[7] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 275–287, Mar. 2007.

[8] M. Luizelli, L. Bays, L. Buriol, M. Barcellos, and L. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 98–106, May 2015.

[9] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Network and Service Management (CNSM), 2014 10th International Conference on*, pp. 418–423, Nov 2014.

[10] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pp. 7–13, Oct 2014.

[11] P. H. Huang, K. W. Li, and C. H. P. Wen, "Nachos: Network-aware chains orchestration selection for nfv in sdn datacenter," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pp. 205–208, Oct 2015.

[12] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Network and Service Management (CNSM), 2015 11th International Conference on*, pp. 50–56, Nov 2015.