On the Latency Benefits of Edge NFV

Richard Cziva, Dimitrios P Pezaros
Networked Systems Research Laboratory, University of Glasgow, UK
r.cziva.1@research.glasgow.ac.uk, dimitrios.pezaros@glasgow.ac.uk

Keywords

Network Function Virtualization, Latency, Edge Network

1. INTRODUCTION

Next-generation networks are expected to support low-latency, context-aware and user-specific services in a highly flexible and efficient manner. Proposed applications include high-definition, low-latency video streaming, remote surgery, as well as applications for tactile Internet, virtual or augmented reality that demand network side data processing (such as image recognition, transformation or head/eye motion aware rendering). One approach to support these use cases is to introduce virtualized network services at the edge of the network, in close proximity of the end users to reduce end-to-end latency, time-to-response and unnecessary utilization of the core network, while providing flexibility for resource allocation. While many research projects including our previous work on Glasgow Network Functions [1] [2] have proposed running virtual network functions (vNF)s at the network edge, a latency-optimal placement allocation has not been presented before for the network edge and therefore the impact on user-to-vNF latency has not been investigated.

In this paper, we formulate a simple vNF placement problem that minimizes end-to-end latency from users to their network functions. We have implemented the problem using Integer Linear Programming (ILP) with the Gurobi solver, and evaluated it with a real topology of a network provider. We use our solution to compare two vNF deployment scenarios over an emulation of a national backbone network: a two-tier edge deployment and a cloud-only deployment. We show that, in our example, using edge servers can deliver up to 70% improvement in user-to-vNF latency.

2. EDGE VNF PLACEMENT

We have defined the 'Edge vNF placement' problem to find the latency-optimal allocation of vNFs. This problem differs from the typical vNF placement problems in the following ways:

1. Our objective is to minimize *end-to-end latency* between all users and their vNFs (instead of minimising the number of vNF servers used [4]).
2. Our placement allocates vNFs to heterogeneous, distributed nodes (e.g, low-cost edge devices or the cloud) with variable latency on the network links.
3. We introduce "latency-sensitive" vNFs that must be placed in a certain latency radius.

To understand our problem in detail, let's denote $\mathbb{N} = \{n_1, n_2, \ldots\}$ to be the set of all vNFs in the network. For each n_i we can define memory, CPU and IO requirements $\{cpu, memory, io\}$, as well as $maxlatency$ that denotes the maximum latency between the vNF and the subscribed user. Let $\mathbb{H} = \{h_1, h_2, \ldots\}$ be the set of vNF hosting devices (that represent either a cloud or an edge server). Similar to vNFs' requirements, each h_i has its own capacity properties $\{cpu, memory, io\}$. Furthermore, let $\mathbb{U} = \{u_1, u_2, \ldots\}$ denote all users. Each u_i has a set of NFs assigned $\{n_1, n_2, \ldots\}$. We also define a latency matrix l calculated by the position of users, servers and the network topology. l_{ij} gives the latency between the user of the n_i vNF in case the vNF is located at h_j.

Finally, let X_{ij} be a binary decision variable that denotes allocations of vNFs to hosts. Hence, X_{ij} is:

$$X_{ij} = \begin{cases} 1 & \text{if } n_i \text{ is allocated to } h_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The *Edge vNF placement problem* is defined as follows: Given the set of users $\mathbb{U}$, set of vNF hosts (edge or cloud servers) $\mathbb{H}$, set of vNFs $\mathbb{N}$ and a latency matrix l, we need to find an appropriate allocation of all network functions that minimizes the total expected end-to-end latencies from all users to its vNFs:

$$\begin{aligned} &\min \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} X_{ij} l_{ij} \\ &s.t. \\ &\quad \sum\nolimits_{n_i \in \mathbb{N}} X_{ij} n_i.requirements < h_j.capacity, \forall h_j \in \mathbb{H} \\ &\quad \sum\nolimits_{h_j \in \mathbb{H}} X_{ij} l_{ij} < n_i.maxlatency, \forall n_i \in \mathbb{N} \\ &\quad \sum\nolimits_{h_j \in \mathbb{H}} X_{ij} = 1, \forall n_i \in \mathbb{N} \end{aligned} \tag{2}$$

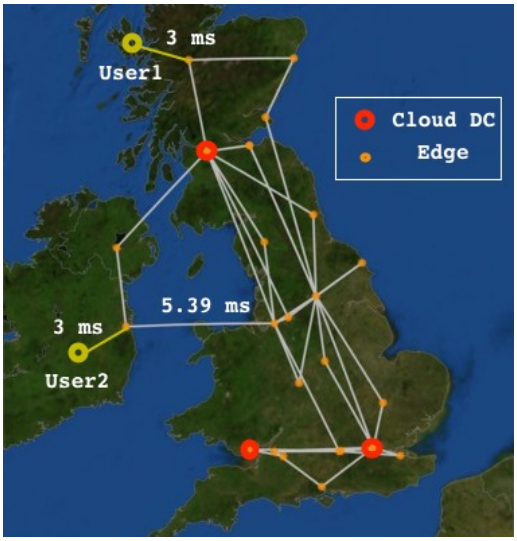

Figure 1: JANET Backbone topology used for our experiments from Topology Zoo. We show two example users connected at two locations with 3 ms latency to the backbone network.

The first constraint ensures that vNFs are placed to servers with sufficient capacity. The second constraint ensures that latency-sensitive vNFs are placed subject to not violating the maximum latency requirement from their users. The last constraint ensures that all vNFs are allocated somewhere exactly once.

3. PRELIMINARY EVALUATION

By using ILP and the Gurobi solver, we can calculate the optimal placement (and latencies) for our problem detailed before. For this paper we have chosen to compare the latency benefits between an edge and a cloud-only vNF deployment scenario. As shown in Figure 1, for our experiments we used a simulation of JANET, the UK NREN backbone as reported by Topology-zoo[1]. We compare two scenarios:

- Cloud-only deployment: vNFs can only be allocated to a set of cloud DCs (three DCs in our case).
- Two-tier edge deployment: in addition to the cloud DCs, all points of presence of the backbone network have equal amount of computing capabilities to host vNFs. When edge devices run out of resources, vNFs get allocated to clouds.

In the simulation, we have assigned users to edge locations with a uniform latency of 3 ms. For all other links we estimated the latency based on multiple parameters (e.g., geographical distance, device latencies, speed of light in fibre, etc.) - as an example, the latency between Glasgow and London in our topology is estimated to be 9.27 ms. End users were assigned uniformly to edge nodes in a round robin fashion. For this experiment we assigned 3 vNFs to each user. For the edge, we have assigned computing capabilities to all edge nodes with a total capacity of 1000 vNFs (around 40 vNFs per edge). The three cloud DCs are considered to have unlimited vNF hosting capacity.

[1]http://topology-zoo.org

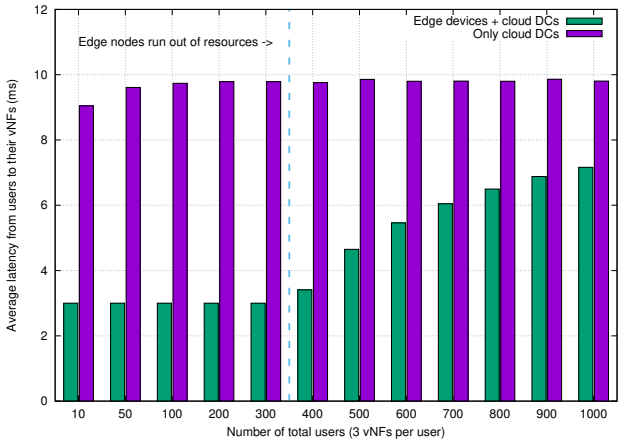

Figure 2: Comparing the average latency from users to their vNFs between edge and cloud vNFs. As shown, when edge nodes fill up, the latency starts increasing and it slowly converges to the latency provided by the cloud vNFs.

In Figure 2, we show the average latency from users to their vNFs. The cloud-only deployment gives an average latency of 10 ms between users and their vNFs, while running vNFs at the edge results in a 3 ms latency (which is in fact the latency from the users to the edge locations) until the edge nodes reach capacity. When edge nodes run out of resources, the average latency between vNFs and the users starts converging to the cloud-only scenario, since now vNFs are being allocated not only at the edge, but also to the cloud servers.

4. CONCLUSION AND FUTURE WORK

In this paper, we have presented a simple vNF allocation problem that optimises latency between users and their vNFs. We have formulated and solved the problem using ILP. To show the benefits of allocating vNFs at the network edge, we have used a simulated nationwide network. We are looking into applying an online version of the presented placement in our Glasgow Network Functions NFV framework[2]. Also, since the presented 'Edge VNF placement' problem is NP-hard (the NP-complete Multiple Knapsack Problem (MPK) [3] can be reduced to our problem), we will be looking for an algorithmic (heuristic) approach to solve it in polynomial time.

5. REFERENCES

[1] R. Cziva, S. Jouet, and D. P. Pezaros. Roaming Edge vNFs using Glasgow Network Functions. In *Proc. of ACM SIGCOMM*, pages 601–602, 2016.

[2] R. Cziva and D. P. Pezaros. Container Network Functions: Bringing NFV to the Network Edge. In *Communications Magazine*. IEEE, June 2017.

[3] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer Verlag, 2004.

[4] H. Moens and F. De Turck. VNF-P: A model for efficient placement of virtualized network functions. In *CNSM*, pages 418–423. IEEE, 2014.

[2]https://netlab.dcs.gla.ac.uk/projects/glasgow-network-functions