Demand-adaptive VNF placement and scheduling with low latency in optical datacenter networks

Tao Gao, Xin Li, Yu Wu, Massimo Tornatore, Biswanath Mukherjee , et al.

Tao Gao, Xin Li, Yu Wu, Massimo Tornatore, Biswanath Mukherjee , Weixia Zou, Shanguo Huang, "Demand-adaptive VNF placement and scheduling with low latency in optical datacenter networks ," Proc. SPIE 11048, 17th International Conference on Optical Communications and Networks (ICOCN2018), 110480E (14 February 2019); doi: 10.1117/12.2518327

SPIE.

Event: 17th International Conference on Optical Communications and Networks (ICOCN2018), 2018, Zhuhai, China

Demand-Adaptive VNF Placement and Scheduling with Low Latency in Optical Datacenter Networks

Tao Gao [a, b], Xin Li [a], Yu Wu [b], Massimo Tornatore [b], Biswanath Mukherjee [b], Weixia Zou [c], and Shanguo Huang [a]

[a]Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing, P.R China; [b]Department of Computer Science, University of California Davis, USA, [c]Key Lab of Universal Wireless Communications, Beijing University of Posts and Telecommunications, Beijing, P.R China

ABSTRACT

We investigate virtual-network-function placement and scheduling problem in optical datacenter networks, considering the installation/de-installation latency of VNF and the rapid variation of low-latency-demands. The proposed scheme achieves low blocking probability, latency, and spectrum resource consumption.

Keywords: Network function virtualization (NFV), placement and scheduling, optical datacenter networks, low latency

1. INTRODUCTION

Recently, network function virtualization (NFV) is emerging as an attractive technology to reduce operational expense (OPEX) and capital expenditures (CAPEX) as well as to provide flexible deployment of new network services [1]. Instead of implementing network functions in hardware middleboxes, which are expensive and hard to maintain and upgrade, NFV enables virtual network functions (VNFs) (e.g., a virtualized firewall or video optimizer) to run on generic commercial off-the-shelf (COTS) servers. Thus, network services can be realized efficiently by routing the traffic through an ordered set of VNFs, which is referred to as a service function chain (SFC) [2]. NFV is recently attracting attention in the context of in elastic optical datacenter networks (EODNs) with the objective of provisioning new services with spectrum efficiency and low-latency [3].

To serve a SFC demand, the problem of VNF placement problem has to be solved, which has stimulated intense research in the last years. In [4], the authors proposed a scheme to solve the VNF placement problem in a network-enabled cloud (NeC) with a varying number of NFV-capable nodes, whose results indicate that a NeC with NFV-capable nodes can significantly decrease the consumption of network resources. In [5] the VNF placement problem was solved in the context of EODNs, by considering joint spectrum and IT resource assignment in a static traffic scenario. In addition to the VNF placement, the VNF scheduling shall be decided, especially for low-latency services. Note that processing latency incurred by each VNF might be different, as each VNF has a specific throughput and the sizes of data to process differ for different service demands. Hence, how to schedule VNFs in a network is important to reduce the latency and resource consumption. In [6], the authors accounted for services with deadlines and formulate the VNF placement and scheduling problem in a static traffic scenario. But, as in a realistic EODN, user demands are usually time-varying [7], the VNFs should be placed and scheduled dynamically according to the network state variation. Further, in a dynamic scenario, it consumes a significant amount of time to install a VNF instance into a virtual machine (VM) or a container [8], which makes users suffer additional latency and increases the operation cost. Thus, it is desirable to avoid frequent installation and de-installation of VNF instances in datacenters. To the best of our knowledge, the problem of VNF placement and scheduling accounting for installation time has not been studied before.

In this study, we focus on the problem of VNF placement and scheduling in a dynamic traffic scenario in EODNs, which can adapt to the variation of service demands. Specifically, in the proposed demand-adaptive VNF placement and scheduling (DA-VNFPS) scheme, each instance will remain in an idle state for a period of time after it finishes any processing task. During the period, if another demand requires this VNF, the instance can start processing data immediately without the installation procedure, which can reduce latency significantly. Moreover, in the proposed scheme, the idle-state duration of VNFs can be adjusted dynamically according to the requirements of service demands to save computing resource, e.g., a VNF with higher popularity will have a longer idle-state duration. Numerical results

17th International Conference on Optical Communications and Networks (ICOCN2018),
edited by Zhaohui Li, Proc. of SPIE Vol. 11048, 110480E · © 2019 SPIE
CCC code: 0277-786X/19/$18 · doi: 10.1117/12.2518327

show that the proposed scheme achieves lower latency and blocking probability (BP) compared with a conventional scheme that is non-adaptive to service demands, while spectrum resource can also be saved.

2. PROBLEM STATEMENT

An EODN is modeled as an undirected graph $G(V, D, E)$, where V denotes the set of optical nodes, D denotes the set of NFV-capable datacenters, and E denotes the set of fiber links. We assume that a fiber link has φ frequency slots (FSs) and the transmission rate per subcarrier is 12.5 Gbps (compatible with employing as modulation format binary phase shift keying (BPSK)). A request is modeled as $R(A, s, d, F, p, l)$, where A denotes the arrival time, s denotes the source node, d denotes the destination node, F denotes the required SFC, p denotes the size of data to be processed, and l denotes the latency requirement, respectively. $F = <f_1, f_2 \ldots f_n>$ is the set of VNFs through which the traffic should be steered in a specified order. The set of all optional VNFs in the network is defined as T. Each VNF requires some computing resources (e.g., CPU cores) and has a specific throughput (e.g., a Network Address Translator (NAT) requires one CPU core for throughput of 5 Gbps [4]). Moreover, for each VNF, the holding time t_{hold} from when the instance is installed in a VM to when it is de-installed from the VM is calculated using $t_{hold} = t_{ins} + t_{pro} + t_{idle} + t_{deins}$, where the items of the right side denote the installation time, the processing time, the idle-state duration, and the de-installation time, respectively. Without loss of generality, we only consider CPU cost c_{total} for computing resource, which is calculated by Eq. (1). In Eq. (1), n_i^f denotes the number of CPU cores consumed by the VM hosting instance i of VNF f and I_f denotes the set of all instances of VNF f.

$$c_{total} = \sum_{f \in T} \sum_{i \in I_f} n_i^f \cdot t_{hold} \tag{1}$$

Note that for each required VNF, the used data will be processed either by the instance that has been installed in a VM or by a new instance that will be installed in a VM, where an installation latency will be incurred. The total latency that a service demand incurs includes: the VNF processing latency, the transmission latency, and the propagation latency.

Fig. 1 illustrates the advantages of the proposed DA-VNFPS scheme (Fig. 1(b)) over a conventional (i.e., "non-adaptive" to demand) VNF placement and scheduling (DNA-VNFPS) scheme (Fig. 1(a)). We consider three requests R_1(0, 3, 2, F_1, 1G, 1.2s), R_2(0.4, 2, 3, F_2, 1G, 1.2s), and R_3(0.8, 3, 2, F_1, 1G, 1.1s). The installation latency for all VNFs is 0.1s and different VNFs have specific CPU core requirements and throughputs. Propagation latency and the procedure of de-installing VNF instances are ignored in the example. As shown in Fig. 1(a), when R_1 arrives in the network, new instances of required VNFs are installed in datacenter d_1, incurring an installation latency. After VNF f_1 processes the data of R_1 (processing latency is 1G/4Gbps=0.25s), the data will be transmitted to VNF f_3 and the processing latency is 1G/2Gbps=0.5s. One FS is allocated for the light paths from node 3 to 1 and from 1 to 2, for which the transmission latency is 1G/12.5Gbps=0.08s. The VNF placement and scheduling strategies for R_2 and R_3 are similar

In the proposed DA-VNFPS scheme as shown in Fig. 1(b), the VNF instances will remain in an idle state for a period of time after they finish processing tasks. Moreover, the duration can be adjusted according to service demands, where a VNF with a higher popularity (i.e., being visited more frequently) will have a longer idle-state duration. For R_1, the VNF placement and scheduling strategy is the same as that in Fig. 1(a). After VNFs f_1 and f_3 finish processing user data, they

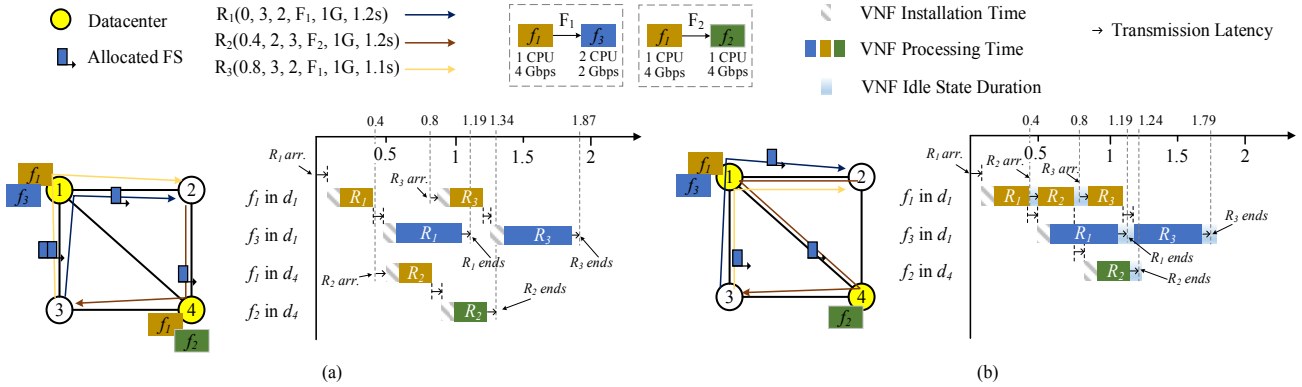

Fig. 1. Different VNF placement and scheduling schemes: (a) DNA-VNFPS, (b) DA-VNFPS.

will remain in the idle state (assume that the initial base idle-state duration is 0.1s). Thus, for R_2, VNF f_1 in datacenter d_1 can start to process its data immediately without a procedure of installation. Since VNF f_1 has been required by service demands twice, which is more frequently than the other VNFs, the idle-state duration is dynamically adjusted to a longer value (0.15s in the example). For R_3, both required VNFs f_1 and f_3 are in the idle state when the user data arrives such that they start to process immediately. Note that, for R_3 in DNA-VNFPS, two FSs are allocated for each light path since the transmission latency constraint is strict while one FS is enough for DA-VNFPS, in which case much spectrum resource is saved. Table 1 compares latency gaps of requests R_2 and R_3 between the DNA-VNFPS and DA-VNFPS schemes, and Table 2 compares the CPU costs between them.

Table 1. Latency Comparison between Two Schemes

Request	*Ins. Lat.*	*Trans. Lat.*	*Proc. Lat.*	*Total Lat.*	*Gap*
R_2-DNA	0.1×2	0.08×3	0.25+0.25	0.94	12%
R_2-DA	0.1	0.08×3	0.25+0.25	0.84	
R_3-DNA	0.1×2	0.04×3	0.25+0.5	1.07	7.5%
R_3-DA	0	0.08×3	0.25+0.5	0.99	

From Table 1, we can see the latency can be reduced significantly by remaining VNF instances in the idle state. In this example, considering CPU core requirements and holding time for all VNFs, the CPU costs of DNA-VNFPS and DA-VNFPS are 3.8 and 4.35 units (core number × second), respectively (see Table 2), calculated by Eq. (1). Although DA-VNFPS consumes a bit more computing resource than the DNA-VNFPS scheme, it reduces the latency for requests as well as spectrum resource consumption significantly. However, a longer idle-state duration will waste the CPU resource and hence the duration should be adjusted dynamically, where a popular VNF should have a longer idle state. The method to calculate the idle-state duration for each VNF according to service demands is introduced in the next section. Further, it is important to note that since the total latency is reduced, the transmission latency constraint is loosened and the allocated FSs can be reduced while the latency requirement is still satisfied.

Table 2. CPU Cost Comparison between Two Schemes

Scheme	*Instance*	t_{hold} (s)	*CPU Core*	*CPU Cost*	*Total Cost*
DNA	f_1 in d_1	0.7	1	0.7	3.8
	f_3 in d_1	1.2	2	2.4	
	f_1 in d_4	0.35	1	0.35	
	f_2 in d_4	0.35	1	0.35	
DA	f_1 in d_1	1.2	1	1.2	4.35
	f_3 in d_1	1.35	2	2.7	
	f_2 in d_4	0.45	1	0.45	

3. METHODOLOGY

To reduce the latency for service demands, while ensuring the CPU cost is not very high, the idle-state duration for each VNF should differ from one VNF to the other and be adjusted dynamically according to its popularity. Specifically, a more popular VNF should have a longer idle-state duration compared to those that are seldom visited. In this study we propose to calculate the idle-state duration as in Eq. (2): where l_f denotes the idle-state duration for VNF f, t_{base} denotes the initial basic duration of the idle state, and t_{scale} denotes the scale of the idle-state duration, v_f denotes the times of VNF f is requested in the network, and v_{total} denotes the total times that all VNFs are requested. If a VNF f is required frequently, which means it has a higher popularity, the value of v_f will become larger and thus it has a longer idle-state duration.

$$l_f = t_{base} + t_{scale} \cdot \frac{v_f}{v_{total}}, f \in T \tag{2}$$

An efficient heuristic algorithm is proposed for DA-VNFPS, whose pseudo-code is reported in Alg. 1. In the algorithm, relevant parameters are first initialized in line 1, where T_i is a time indicator and S_i is the set of VNF instances to serve the service demand. From line 2 to 5, the qualified VNF instances are obtained, which are available at the time estimated in terms of processing latency. In lines 6-20, for each required VNF, either an installed instance is selected or a new instance is installed, whose location should be near to that the last VNF instance is hosted. In line 21, number of FSs required is calculated according to the latency requirement and the data size. Specifically, assuming that the maximum

Algorithm 1 DA-VNFPS

Input: SFC request R_i (A, s, d, F, p, l)
Output: The VNF placement and scheduling solution for R_i

1. Initialize the time indicator $T_i = A$, the instance set S_i to serve R_i;
2. **for** each VNF f in the required SFC F **do**
3. Obtain the available instances of VNF f at time T_i, and add it to candidate set C_f for VNF f;
4. Estimate the processing time t_{proc} according to the throughput of f and the size of user data to be processed, update T_i=T_i+t_{proc};
5. **end for**
6. **for** each C_f **do**
7. **if** f is not the first VNF in F **do**
8. **if** $C_f \neq \varnothing$ **do**
9. Choose the instance of VNF f, of which the location is nearest to prior VNF' location, and add it to S_i;
10. **else**
11. Choose the datacenter where prior VNF is installed and install a new instance of VNF f to it. Add it to S_i;
12. **end if**
13. **else**
14. **if** $C_f \neq \varnothing$ **do**
15. Choose the instance of VNF f, whose location is nearest to the source node, and add it to S_i;
16. **else**
17. Choose the datacenter nearest to the source node and install a new instance of VNF f to it. Add it to S_i;
18. **end if**
19. **end if**
20. **end for**
21. Calculate the number of FSs n required to ensure the latency requirement;
22. **for** k = 1 to $|S_i|$ -1 **do**
23. Calculate the light-path between the datacenters hosting f_k and f_{k+1}, allocate n continuous FSs using the First Fit algorithm;
24. Calculate the duration value l_f of the idle state for VNF f_k according to the visit times by service demands using Eq. (2);
25. **end for**
26. Calculate the duration value for VNF $f_{|s|}$ with Eq. (2);
27. Check whether latency requirement is satisfied;
28. **if** satisfied **do**
29. Update the idle state duration for each VNF in S_i;
30. **else**
31. Release resources allocated for R_i, mark it blocked;
32. **end if**

transmission latency can be tolerated is M seconds and the size of data to be processed is p Gbits, then number of required FSs $n = \lceil p/(12.5 \cdot M) \rceil$. From line 22 to 26, light-paths are calculated between the locations where two consecutive VNF instances are hosted and FSs are allocated for each light-path. Also, the idle-state duration is calculated by Eq. (2) for each VNF according to the service demand. Finally, in lines 27-32, whether the latency requirement of the service demand is satisfied is checked and the idle-state duration is updated for each VNF. For DNA-VNFPS, the difference from Alg. 1 exists in lines 22-26, where the DNA-VNFPS does not have a procedure of calculating idle-state duration.

4. NUMERICAL RESULTS

We evaluate the performance of the proposed DA-VNFPS scheme in terms of BP, average latency, and resource utilization (including spectrum resource and CPU resource) on a 28-node US Backbone network [9]. Arrivals of SFC demands follow a Poisson distribution with λ requests per second and the size of user data is uniformly distributed in a range of [2.5, 5, 7.5, 10] Gbits. We assume each SFC is composed of four kinds of VNFs of which required CPU cores are in a range of [1, 4] and the throughputs are 1, 2, 3, 5 Gbps, respectively. To obtain precise results, we run the simulation program 100 times at every traffic load scenario and take the average. Each run contains 10,000 requests. Results of BP and average latency are shown with a margin of error at a confidence interval of 95%.

1) Blocking probability: Fig. 2(a) illustrates the numerical results for DA-VNFPS (with different values of t_{base} and t_{scale}, denoted by *"-t_{base}/t_{scale}"*) and DNA-VNFPS. We can find that DA-VNFPS-4/4 reduces BP by about 28% at the traffic load of 800 Erlang than DNA-VNFPS. Moreover, DA-VNFPS-0/4 and DA-VNFPS-4/0 schemes achieve a little higher BP compared with DA-VNFPS-4/4, which means a longer duration of the idle state can help to improve the BP performance. However, there should be a trade-off between the idle state duration and CPU resource consumption, since a longer idle-state duration will increase the CPU cost.

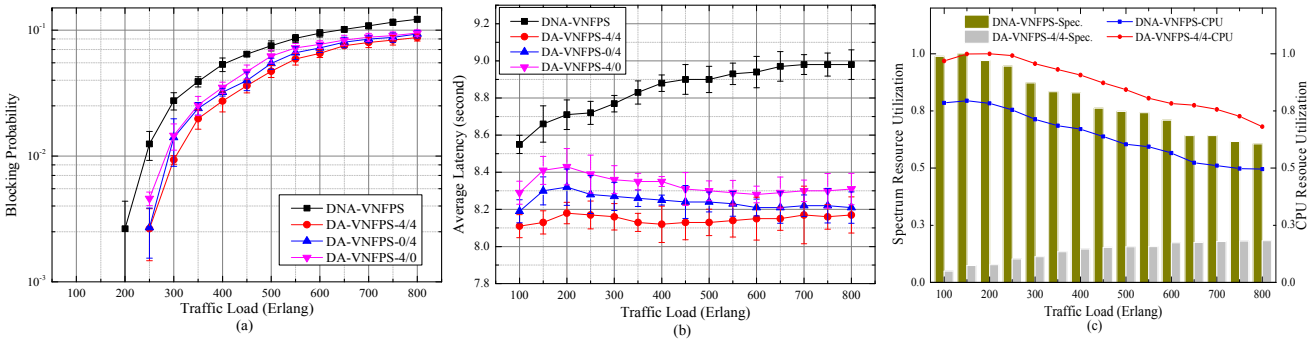

Fig. 2. Simulation results: (a) Blocking probability, (b) Average latency, and (c) Resource utilization.

2) Average latency: With respect to the average latency as shown in Fig. 2(b), the reduction between DA-VNFPS-4/4 and DNA-VNFPS can reach about 10%. A variation of the values of t_{base} and t_{scale}, does not significantly affect DA-VNFPS results, even if it can be noticed that DA-VNFPS-0/4 achieves lower average latency compared with DA-VNFPS-4/0. This indicates that adjusting the duration of the idle state for the VNF according to its popularity can achieve better performance than a constant duration for each VNF. As different values of t_{base} and t_{scale} have an important effect on the performance in terms of BP, latency, etc., a more accurate study on how to decide these parameters is left as future work.

3) Resource utilization: In Fig, 2(c), we plot the spectrum resource utilization and the CPU resource utilization for different schemes, normalized by their maximum value. With the increase of traffic load, both CPU and spectrum resource consumption reduce for DNA-VNFPS, because more demands are blocked at high traffic load. Moreover, DA-VNFPS allows to reduce the spectrum consumption by up to 98%, but it has a slightly higher CPU resource consumption compared to DNA-VNFPS. The reason why the reduction of spectrum resource consumption between DA-VNFPS and DNA-VNFPS is huge consists of two aspects: i) for DNA-VNFPS, to satisfy the strict latency requirement of user demands, the data is more likely to be transmitted between different active VNF instances distributed in different datacenters to avoid the installation latency. In this context, extra spectrum resource is consumed to transmit data. However, in DA-VNFPS, instances of different VNFs in the idle state are more likely to be hosted in a single datacenter, which saves spectrum resource usage; ii) in DNA-VNFPS, the installation latency has an important effect on the total latency, making the transmission latency constraint much stricter than that in DA-VNFPS. Hence, to satisfy the latency requirement of service demands, more spectrum resource is allocated to reduce the transmission latency.

5. CONCLUSION

NFV promises to enable a more flexible deployment of SFC in EODNs. To achieve this objective, it is important to place and schedule the VNFs of a SFC according to the service demands. In this study, we focus on the problem of dynamic VNF placement and scheduling in EODNs, where an instance of a VNF is allowed to remain in an idle state for a period of time in terms of its popularity. Numerical results indicate that the proposed scheme can reduce the latency, the spectrum resource consumption, and the BP significantly compared with a conventional scheme.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (Nos. 61701039, 61331008, 61601054 and 61571058), and the National Science Foundation for Outstanding Youth Scholars of China (No.61622102). B. Mukherjee's and M. Tornatore's work is also supported by U.S. National Science Foundation Grant No. 1716945.

REFERENCES

[1] Network Functions Virtualization (NFV) Architectural Framework, ETSI GS NFV 002 V1.1.1, Oct. 2013.
[2] Rashid Mijumbi, et al., "Network function virtualization: State-of-the-art and research challenges," IEEE Comm. Surveys & Tutorials, vol. 18, no. 1, pp. 236-262, 2016.
[3] M. Xia, et al., "Optical Service Chaining for Network Function Virtualization," IEEE Comm. Mag., vol. 53, no. 4, pp. 152–158, 2015.
[4] A. Gupta, et al., "On service-chaining strategies using Virtual Network Functions in operator networks," vol. 133, pp. 1-16, 2018.
[5] W. Fang, et al., "Joint Spectrum and IT Resource Allocation for Efficient VNF Service Chaining in Inter-Datacenter Elastic Optical Networks," IEEE Commun. Letters, vol. 20, no. 8, pp. 1539-1542, 2016.
[6] H. Aessem, et al., "Scheduling Service Function Chains for Ultra-Low Latency Network Services," in Proc. 13th Int. Conf. Netw. Service Manage (CNSM), pp. 1-9, 2017.
[7] T. Gao, et al., "User Demands-Adaptive Dynamic Content Replacement in Elastic Optical Datacenter Networks", in Proc. ACP, 2016.
[8] A. Sheoran, et al., "An Empirical Case for Container-driven Fine-grained VNF Resource Flexing," in Proc. IEEE Conference on Network Function Virtualization and Software Defined Networks, 2016.
[9] T. Gao, et al., "A Shared Segment Protection Approach for Distributed Sub-Tree Based Optical Multicasting Scheme in Elastic Optical Datacenter Networks," in Proc. OFC, pp. Th2A.37, 2018.