DAVIS: A Delay-Aware VNF Selection Algorithm for Service Function Chaining

Gaurav Garg
Department of CSE
IIT Hyderabad, India
cs16mtech11022@iith.ac.in

Venkatarami Reddy
Department of CSE
IIT Hyderabad, India
cs17resch01007@iith.ac.in

Antony Franklin A
Department of CSE
IIT Hyderabad, India
antony.franklin@iith.ac.in

Bheemarjuna Reddy Tamma
Department of CSE
IIT Hyderabad, India
tbr@iith.ac.in

***Abstract*—The mobile network operators are moving towards Virtual Network Functions (VNFs), in place of traditional dedicated hardwares for the 5G deployment. The VNFs require low maintenance and give high flexibility compared to dedicated hardwares which are prone to failures. Typically, VNFs are run on top of Virtual Machines (VMs) which run on physical nodes in data centers. For each class of service, based on its policy, the traffic has to pass through a set of VNFs in sequence termed as Service Function Chains (SFCs). Selection of available VNFs to steer the SFC requests with guaranteed end-to-end latency is still an open problem. In this work, we design an algorithm named DAVIS (Delay-Aware VNF selectIon algorithm for SFC) to increase the SFC acceptance rate by efficiently selecting VNFs. Simulation results show that DAVIS improves the throughput of accepted requests by 42% while guaranteeing SLA requirements.**

***Index Terms*—Network Function Virtualization, Service Function Chain, Processing Latency, CPU Utilization.**

I. Introduction & Motivation

Service Function Chaining (SFC) is an ordered set of Network Functions (NFs) which guides traffic through them to provide different end-to-end services. In traditional networks, each NF is built on dedicated hardwares which are deployed and maintained manually at fixed locations based on requirements. Due to the rapid explosion of mobile and IoT devices, the amount of traffic generated from users is increasing significantly [1]. To handle such a massive increase in the user traffic, network operators need to install more dedicated hardwares. However, this approach is not efficient, concerning the installation time and cost. With the emergence of technologies such as Network Functions Virtualization (NFV) and Software Defined Networking (SDN), it is possible to dynamically create virtualized instances of NFs (*i.e.,* Virtual Network Functions (VNFs)). These VNFs, in turn, interconnect to form SFCs which efficiently utilize the resources, thereby lowering the OPEX (OPerational EXpenditures) and CAPEX (CAPital EXpenditures) [2]. For instance, if a service deployed by an operator requires traffic to pass through the firewall, load balancer, and then WAN acceleration, an SFC will be formed if and only if the flow of traffic happens in such a manner [3].

Based on the type of services and its policies, orchestrator needs to select the required VNFs and steers traffic through it. In order to meet the traffic demands of different services, an operator can deploy multiple instances of the same VNF type at different locations in a large-scale telecom network. To improve resource utilization and SFC acceptance rate, an operator can also select the same VNF instance for multiple services. An efficient selection of VNFs to form SFCs by considering both link and processing delays while meeting the end-to-end delay with less service interruption in its lifetime is an important aspect to be addressed.

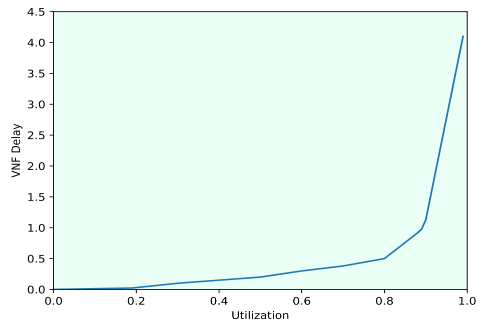
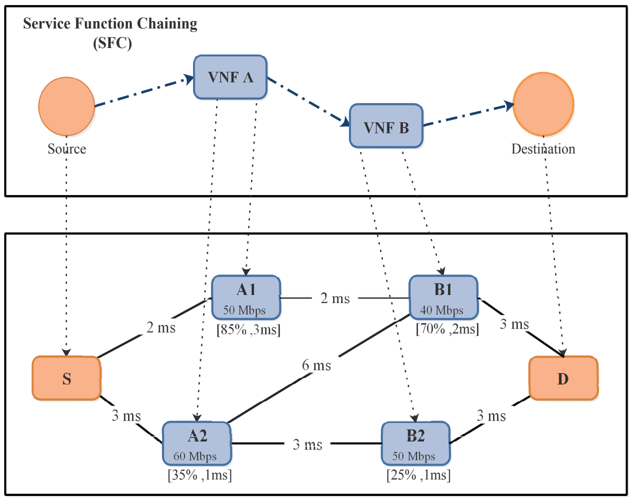


Figure 1: CPU Utilization vs VNF Delay.

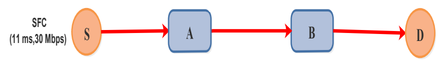
Numerous studies on VNF selection to form SFC considers link delays and/or throughput of the VNFs [4] [5]. In [6], the authors showed the relationship between resource allocated to VNF instance and its expected latency. However, all these works do fail to consider processing delay of VNFs which changes with the CPU utilization.

As the number of accepted SFC requests increases, CPU utilization of the VNF increases and SLAs may get violated. This may lead to frequent service remappings, which can be severe for some critical applications. Hence in our work, we consider the dependency of VNF delay on CPU utilization, which avoids the service interruption and increases the SFC acceptance rate. The novelty of our work lies in considering the relationship between VNF delay and CPU utilization. In this way, we ensure delay-awareness to create SFC. The graph in Fig. 1 shows the relationship between processing latency of the VNF and CPU utilization. These characteristics change with every VNF. Therefore, the processing latencies at various utilizations end up differently for a firewall, a Load Balancer, and a proxy server [7].

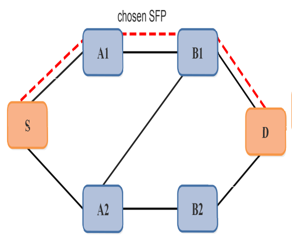
The example given in Fig. 2 illustrates our motivation. SFC



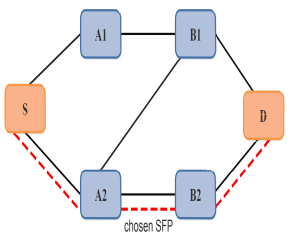
(a) Network Topology.



(b) SFC Request.



(c) Shortest Path Algorithm.



(d) Proposed DAVIS Algorithm .

Figure 2: Working Example for SPA and DAVIS.

considered in this example needs to be processed by VNF A and VNF B in order to meet the end-to-end latency of 11 ms with the required bandwidth of 30 Mbps. In the network, two instances of VNF A and VNF B are deployed at different nodes. The available throughput of each VNF is given at each node. The transmission delay between VNFs is given at link and the format $[u\%, d]$ at the node tells the processing delay d at the CPU utilization at $u\%$ for that VNF. For example, A1 $[85\%, 3\ ms]$ tells that processing delay of A1, which is running at 85%, is 3 ms. Fig. 1(c) shows the Service Function Path (SFP) selected by using the shortest path SFC steering scheme, which prefers to choose the path with minimum end-to-end latency without considering processing delay of VNFs. We call this base scheme as the SPA (Shortest Path Algorithm) scheme, which chooses the path S $\rightarrow$ A1 $\rightarrow$ B1 $\rightarrow$ D with the end-to-end latency of 12 ms which exceeds the tolerable latency and hence the request will be rejected. The proposed approach DAVIS (explained in Section II) considers the processing delay also, chooses the path S $\rightarrow$ A2 $\rightarrow$ B2 $\rightarrow$ D with the end-to-end latency of 11 ms and the request will be accepted.

The objective of this work is to provide an efficient approach for selecting VNFs while maximizing the SFC acceptance rate and the throughput of accepted requests leading to an increase in provider's revenue.

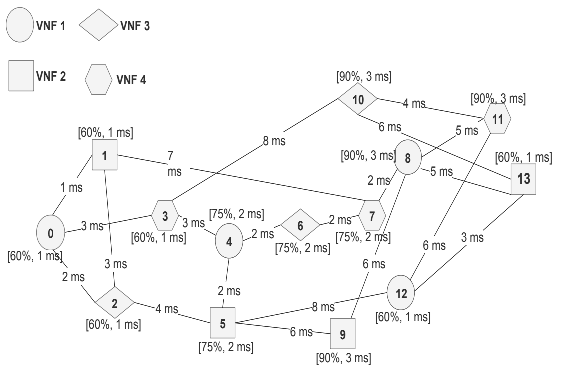


Figure 3: NSFNET Topology.

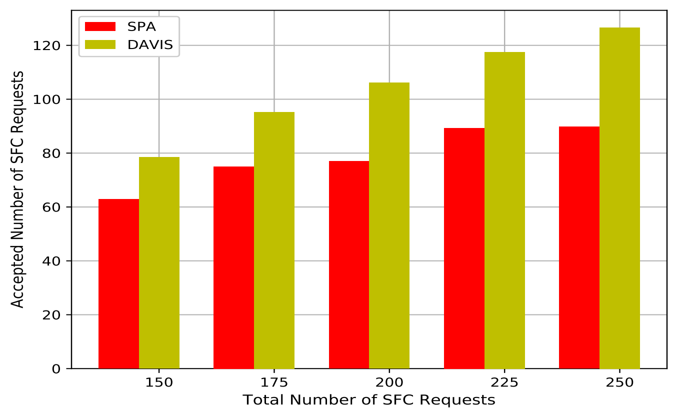
II. Algorithm Design

An instance of each type of VNF is assigned a static CPU utilization threshold. For the assigned threshold of VNF, the processing delay can be known from Fig. 1. $[u\%, t]$ in Fig. 3 means that when the VNF's CPU utilization threshold is set to $u\%$, the maximum processing latency offered will be t ms. Based on the significant change in delay based on CPU utilization, instances have been assigned a certain threshold. By setting the CPU utilization threshold, we guarantee that the VNF will not accept any more SFC requests if the current utilization is equal to the threshold. By doing this, we assure that all the SFC requests passing through this VNF will never have processing latency more than t ms. For example, $[60\%, 1\ ms]$ for $node1$ means that $node1$ stops accepting any requests after the utilization of the CPU reaches 60%, therefore, VNF delay of $node1$ will not exceed 1 ms at any

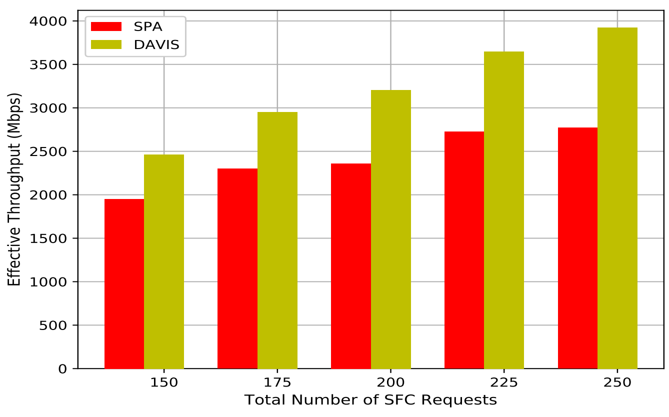
Algorithm 1 DAVIS Algorithm

```
    Input SFC request R, Graph G(V,E)
    Output Path P for each request r ∈ R
 1: Set $Th$ = 0
 2: For each request r ∈ R
 3:         L = length of r
 4:         Convert G into an L-stage graph
 5:         $\{f_1, f_2, ...., f_L\} \leftarrow$ set of ordered VNFs in r
 6:         $f_0 \leftarrow$ source, $f_{L+1} \leftarrow$ destination
 7:         total_delay($f_{L+1}$) = $\min_{f_L \in all\ nodes\ at\ level\ L}$ (
            processing_delay(($f_L$)) + link_delay(($f_{L+1}$),($f_L$))
            + total_delay($f_L$) )

 8:         $lat \leftarrow$ total_delay($f_{L+1}$)
 9:         if $lat \leq tolerable\ latency$
10:                 Accept the request
11:                 $Th \leftarrow Th + bandwidth$
12:                 Update the resources in the path
13:         else
14:                 Reject the request
```



(a) Accepted Number of SFC Requests.



(b) Effective Throughput.

Figure 4: Simulation Results.

point of time.
Algorithm 1 gives an overview of the proposed algorithm DAVIS (Delay-Aware VNF selectIon algorithm for SFC) which considers the relationship between VNF delay and CPU utilization, along with link delays while choosing SFPs.

The input to the algorithm is the SFC request set and the graph which contains the nodes and edges which satisfy the requested VNF throughput and link bandwidth. To find the path with minimum end-to-end delay from source to destination, the graph is represented as a multistage graph and solved by using the dynamic programming approach. We compute the total link delay between 2 nodes of different stages using Dijkstra's algorithm. In line 1, we set Th to be 0. Th is the sum of bandwidths of all accepted requests. For each SFC request, we compute the shortest path from source to destination, such that each VNF and each link on the path satisfy the bandwidth requirement (line 7). The request is accepted if the latency is less than or equal to the tolerable latency of the service. Throughput and resources on the path P are then updated (lines 11-12).

III. Performance Evaluation

A. Simulation setup

Table I: Simulation Parameters.

Link bandwidth	1700 Mbps
Node bandwidth	1500 Mbps
Length of SFC requests	1 - 4 [uniformly distributed]
SFC bandwidth	10 - 50 Mbps [uniformly distributed]
End-to-end Latency	15 - 25 ms [uniformly distributed]

We test the algorithms on a telecom network topology, NSFNET topology (14 nodes and 21 links) as shown in Fig. 3. The delays between the links and the processing delays for the VNFs have been specified in the figure itself. We assume that there are 4 types of VNFs spanned across the 14 nodes of the network. Each node can run only 1 type of VNF. Simulation parameters considered are shown in Table I. The values of some parameters are uniformly distributed in a specific range as mentioned. The length of an SFC request is uniformly distributed and each SFC request can traverse at most 4 VNFs.

To evaluate the proposed approach, we consider the number of accepted SFC requests and effective throughput as the performance metrics. We compare these metrics by running the base scheme SPA and the proposed approach DAVIS.

B. Simulation Results

1) Accepted number of SFC requests: Fig. 4(a) shows the comparison of the accepted number of SFC requests for both the algorithms. We can clearly see that the proposed approach DAVIS performs better than the base scheme SPA. It is because DAVIS considers processing delay also while choosing SFP. So, it selects the VNF such that even if the utilization of the VNF reaches maximum (up to threshold), the SFC does not violate its SLA. The proposed algorithm DAVIS accepts 41% of more SFC requests than the base scheme SPA.

2) Effective throughput: Effective throughput is the sum of throughput of all the accepted requests. Fig. 4(b) shows the comparison of effective throughput for both the algorithms. Similar to accepted number of SFC requests, effective throughput of accepted requests is higher in the proposed algorithm even for larger number of SFCs. The proposed algorithm DAVIS improves the throughput of accepted requests by 42%. Thus, the proposed algorithm DAVIS outperforms the SPA scheme.

IV. Conclusion and Future Work

In this poster, we propose an effective VNF selection algorithm DAVIS to deploy SFCs by considering dependency between processing delay and the CPU utilization along with link delay. Results show that the proposed approach DAVIS performs better than the base SPA in terms of accepted number of SFC requests and effective throughput by upto 41% and 42% respectively.
In the future, we plan to work on calculating the processing latency for different CPU utilization for custom VNFs and evaluating the proposed approach DAVIS for custom VNFs.

V. Acknowledgements

This work was supported by the research project "Low Latency Network Architecture and Protocols for 5G Systems and IoT" funded by Science and Engineering Research Board (SERB), Government of India.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022." Cisco White Paper, November 2018.

[2] E. Halpern, J. and E. C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, DOI 10.17487/RFC7665, October 2015.

[3] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. "SIMPLE-fying Middlebox Policy Enforcement using SDN," in Proc. of ACM SIGCOMM, pp. 27–38, August 2013.

[4] C. Ghribi, M. Mechtri, D. Zeghlache, "A Dynamic Programming Algorithm for Joint VNF Placement and Chaining," in Proc. of ACM Workshop Cloud-Assisted Networking, pp. 19-24, December 2016.

[5] S. Jiao, X. Zhang, S. Yu, X. Song and Z. Xu, "Joint Virtual Network Function Selection and Traffic Steering in Telecom Networks," in Proc. of IEEE GLOBECOM, pp. 1-7, December 2017.

[6] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware VNF Placement and Chaining based on a Flexible Resource Allocation Approach," in Proc. of 13th International Conference on Network and Service Management (CNSM), IEEE, pp. 1–7, November 2017.

[7] F. C. Chua, J. Ward, Y. Zhang, P. Sharma, and B. A. Huberman, "Stringer: Balancing Latency and Resource Usage in Service Function Chain Provisioning," in IEEE Internet Computing, pp. 22-31, November - December 2016.