2022 IEEE 47th Conference on Local Computer Networks (LCN) | 978-1-6654-8001-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/LCN53696.2022.9843704

Prediction-based SFC Placement with VNF Sharing at the Edge

Amir Mohamad*[§] and Hossam S. Hassanein*
*Queen's University, Canada, [§]Cairo University, Egypt
Email: a.mohamad@queensu.ca, hossam@cs.queensu.ca

***Abstract*—The demand for ultra-low latency requirements is fueled by the growing popularity of time-sensitive applications including virtual, augmented and mixed reality, and industrial IoT. Edge computing is positioned to fulfill such stringent latency requirements. Addressing the increasing demand for time-sensitive applications becomes challenging due to limited resource at the edge. Even though virtual network function (VNF) sharing is known to improve the utilization of the service providers' resources, service requests -including time-sensitive ones- can nevertheless be rejected. This paper proposes PSVS: a Prediction-based Service placement scheme with VNF Sharing at the edge. PSVS utilizes the predicted required resources in a defined lookahead window to minimize the rejection rate of premium services. A safety-margin is empirically-defined and used to add resiliency against prediction errors. Results show more than a 50% reduction in the rejection rate of premium services. Moreover, PSVS is resilient to prediction errors.**

***Index Terms*—Edge Computing, SFC, NFV, VNF**

I. Introduction

Communications service providers (CSPs) are revolutionizing the way in which they build and operate their networks by disaggregating and virtualizing conventional proprietary functions both in the core and radio access network (RAN). At the core side, control and user plane separation (CUPS) is introduced to evolve, provision, and scale control and user functions separately. To host the user-plane functions, a distributed cloud infrastructure is required near end users. This goes hand-in-hand with the service-based architecture (SBA) adopted by 3GPP for the 5G core and the adoption of cloud-native platforms and tools [1]. The same distributed cloud is required to host the open RAN (O-RAN) disaggregated and virtualized components: radio unit (RU), distributed unit (DU), centralized unit (CU), and near real-time RAN intelligent controllers (nRT-RICs) [2]. The disaggregated RAN functions must be provisioned with stringent time constraints [3].

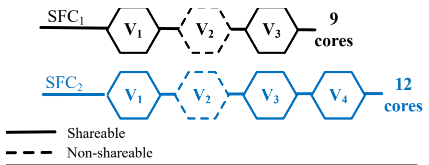
The majority of emerging 5G use cases are time-sensitive/critical in nature, such as real-time media (augmented reality (AR) and virtual reality (VR)), industrial control, remote control, and mobility automation [4]. Time-sensitive, henceforth premium (Pr), services and applications are a class of software that have stringent time constraints and a service would fail if such constraints were not met [5]–[7]. For applications with extremely limited time budget, catastrophic consequences might follow as a result of service failure, for example, a collision warning service failure might result in more collisions and more fatalities.

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: RGPIN-2019-05667.

Edge computing, especially multi-access edge computing (MEC) is the distributed cloud to fulfill the ultra-low latency and high-reliability requirements of use cases mentioned earlier. As a result, interest from industry in edge computing has grown substantially. CSPs are capitalizing on edge computing to host their services, core and RAN virtualized network functions (VNFs), in addition to third-party services, including over-the-top (OTT) services. The edge computing market is projected to grow from $3.6 billion in 2020 to $15.7 billion by 2025, at a growth rate of 34.1% during the forecast period [8].

By adopting network function virtualization (NFV) and software-defined networking (SDN), service provisioning is more agile, scalable, and costs less as CSPs can reduce capital and operations expenditures [9]. To provision a service, an orchestrator has to take a placement decision to deploy service functions (VNFs) on hosting physical nodes. Since the introduction of NFV in 2012, there has been a large body of research addressing VNF placement and resource allocation. Most enterprise and network services consist of component functions/VNFs forming service function chains (SFCs) and traffic should traverse these functions in a specific order. In Figure 1, there are two SFCs with different number of VNFs and the total number of CPU cores required per each SFC. SFCs may have common VNFs, for example sfc_1 and sfc_2 have V_1, V_2 and V_3 common VNFs.

Edge resources are limited compared to the abundant cloud resources. To address the increasing demand for edge computing, efficient utilization of edge resources will play an indispensable role in SFC placement. Time-sensitive premium (Pr) SFCs cannot tolerate waiting for resources to be deployed. If a Pr SFC (sfc_{pr}) request is



VNFs	V₁	V₂	V₃	V₄
Required Cores	5	3	1	3

Fig. 1. Example SFCs of different lengths and common VNFs

received and no resources are available, the request will be rejected. Conversely, lower-priority best-effort (BE) SFC (sfc_{be}) requests can tolerate waiting for resources availability. Prioritizing sfc_{pr} requests over sfc_{be} requests will help in slightly reducing the rejection rate of sfc_{pr} requests. However, such prioritization will not significantly reduce or even eliminate sfc_{pr} requests rejection.

Hence, we propose a prediction-based SFC placement scheme with VNF sharing (PSVS) to reduce the rejection rate of sfc_{pr} requests. Taking advantage of shareable common VNFs among SFCs and the predictability of sfc_{pr} requests arrival, PSVS will decide to satisfy sfc_{be} requests pending deployment or to defer the deployment to save resources for future sfc_{pr} requests. PSVS uses a safety-margin to mitigate the unavoidable prediction errors.

The main contributions of this paper are:

- Introduction of a prediction-based placement scheme that significantly reduces the rejection rate of sfc_{pr} requests.
- Introducing the *safety-margin* that provides the desirable resiliency against prediction errors including extreme errors at high error rate/probability.

The remainder of this paper is structured as follows. Section II covers related work. Proposed prediction-based SFC placement, system model, and problem formulation are described in Section III. Section IV, details the simulation framework, performance evaluation and results analysis. Conclusions and future work are presented in Section V.

II. Related Work

Unlike the placement of single VNFs, the placement of SFC is a two-step process. First, SFC's VNF components are associated with hosting nodes then resources are allocated for each VNF to satisfy performance requirements. Second, forwarding/traffic steering rules are defined to enforce the orderliness of traffic processing within an SFC. SFC/VNF placement and/or resource allocation can be done at different levels. The work in [10] proposes resource allocation at the CPU core level in many-core architectures. At one deeper level, authors of [11] allocate CPU at the time/share and deal with degradations due to operation dynamics.

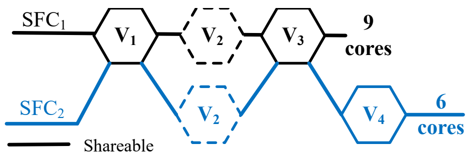


Fig. 2. Resource savings when sharing VNFs

Indeed, "*VNF sharing*" is one of the techniques used to reduce the deployment cost by efficiently utilizing resources. As shown in Figure 2, the required resources to satisfy sfc_1, sfc_2 are 21 $CPU\,cores$ compared to only 15 $cores$ when VNF sharing is used. Unlike some surveyed VNF sharing papers which consider all VNFs are shareable, in this example having V_2 as non-shareable, the VNF sharing is able to use 29% fewer resources. There is an increasing interest in *VNF sharing* among CSPs and OTT service providers. For example, sharing non-security-critical VNFs such as mobility management across end-to-end 5G slices is proposed by [12]. In [13], authors proposed sharing the same cache VNF (vCache) among ISPs with a common infrastructure. Consequently, *VNF sharing* can play an imperative role in reducing the cost of service provisioning by efficiently utilizing resource-limited edge environments. Idle resources and fragmentation are unwanted consequences when not sharing VNFs [14]. Unlike the work in [15], sharing VNF among SFC should consider operation dynamics and refrain from specifying a predefined number of flows that a VNF can serve, which may leave some VNFs underutilized. In the same vein, concerning priority-based SFC/VNF placement, unlike the work in [16], [17], we believe that SFC request priority should be the same for all SFC's VNFs, the priority should not change, and it should be known before satisfying the SFC request.

To better serve customers, service providers utilize prediction in different aspects. Service provider can use models/tools to predict service arrivals which can help reserve resources for Pr SFCs to avoid rejection/failures [18]; traffic demand and user mobility that both help with proactive replication and/or migration [19], [20]; the resources to allocate to guarantee performance requirements (profiling) [21]; finally, to predict the degradation and/or failure of physical nodes hosting SFC's VNFs that help mitigate service interruptions and downtime.

In this paper, we utilize service requests arrival predictions to reduce the rejection rate of premium services. In addition, we utilize a *VNF sharing*-based SFC placement that considers operation dynamics when taking VNF sharing decisions and prioritizes premium over best-effort services. PSVS, the proposed scheme, considers one priority for all SFC components and that priority is known prior to receiving SFC requests.

III. System Model and Problem Formulation

More than two service categories/priorities could have been considered; however, we decided to utilize two quality of service (QoS) categories for simplicity, premium and best-effort. Premium service is provisioned with highest expected load, not oversubscribed, and served with dedicated high-priority queues; and best-effort service is sent and queued with lower priority. Findings in this paper are applicable to service domain with more than two service categories.

Due to operation dynamics, traffic processed by SFC's VNFs vary and VNFs can be underutilized at times. *VNF sharing*-based SFC placement scheme takes advantage of operation dynamics and shareable VNFs to enhance resource utilization, reduce SFC deployment cost and rejection rate. To satisfy a new SFC request, the placement scheme searches running VNFs for similar underutilized VNF which can serve the required load of peer VNF of SFC requests being satisfied. The scheme will only instantiate a new VNF when: considered VNF is non-shareable; there are no similar previously deployed VNFs or a similar VNF(s) deployed but fully utilized. Our work

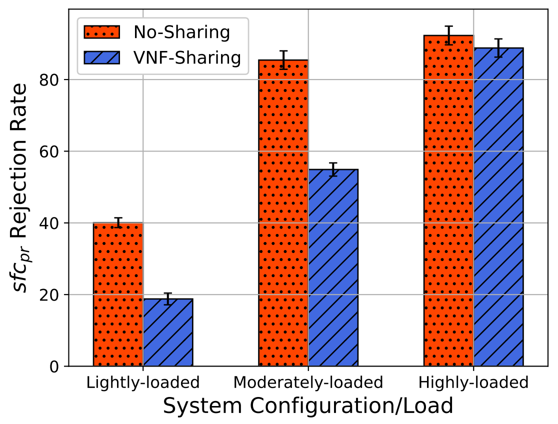
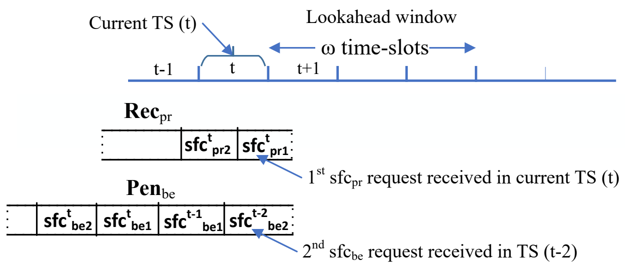


Fig. 3. Rejection rate of Premium (Pr) SFCs using VNF non-sharing vs *VNF-sharing* for different system configurations/loads (Experiment duration is 200 TSs, arrival rate λ = 2 sfc requests/TS) [22]

PSVShare [22], is a scheme for priority-based placement with VNF sharing at the edge. According to the results achieved from our extensive experiments of PSVShare, SFC placement with *VNF sharing* reduced the rejection rate of sfc_{pr} requests, but it is still concerning. As shown in Figure 3, the best-case scenario is *'lightly-loaded'*. Using *VNF sharing*-based placement results in a 50% reduction in sfc_{pr} requests rejection rate. However, the rejection rate of the *VNF sharing*-based placement is still about 19%. It is worse for higher system loads, with longer duration and/or higher number of sfc_{be} requests. These rejections lead to unsatisfied customers and lost revenue for CSPs.

To address such concerning rejections, we propose a prediction-based SFC placement scheme with VNF sharing (PSVS) to help CSPs improve the edge resource utilization and minimize lost revenue opportunities. To minimize the situations where resources are not available to satisfy sfc_{pr} requests, PSVS will utilize predicted sfc_{pr} requests to arrive in a lookahead window of length ω to decide to satisfy the pending sfc_{be} requests or not. To the best of our knowledge, prediction-based SFC placement has never been used in the context of SFC placement with *VNF sharing* at the resource-limited edge environments.



- **Rec$_{pr}$** : a queue holding sfc$_{pr}$ requests received in current time-slot (TS)
- **Pen$_{be}$** : a queue holding sfc$_{be}$ requests received in current TS plus those requests received in previous TSs but never deployed due to resource limitations

Fig. 4. PSVS lookahead window, different queues (Rec_{pr} and Pen_{be}), and queued sfc_{pr}/sfc_{be} requests.

As shown in Figure 4, in each time slot (TS), arriving sfc_{pr} and sfc_{be} requests are placed in Rec_{pr} and Pen_{be} queues, respectively. While sfc_{be} requests can tolerate waiting for deployment if not satisfied in the same TS they were received, the sfc_{pr} requests can not tolerate waiting and are rejected if not satisfied (in Figure 4, unlike sfc_{be} requests in Pen_{be}, there are no sfc_{pr} requests older than current TS in Rec_{pr}). We assume that a simple prediction technique/model exists that can predict the sfc_{pr} requests to arrive in the next ω TSs and the required resources with 100% accuracy (in Section IV, we demonstrated the performance under different prediction errors and error rates). It is known that the longer the lookahead window, the less accurate the predictions, which is why we experimented with $\omega \in [1, 3]$ TSs. The actual duration in seconds of each TS is variable and depends on the number of requests that need to be deployed.

A. System Model

Each SFC requests consists of a list of VNFs. SFC's VNFs are selected from a list of on-boarded VNFs V. Each VNF type $v \in V$ has resource requirements like $CPU\ cores$ and $memory$. A VNF is expected to operate at maximum capacity/throughput, $F_{max}(v)$, if assigned the required resources. As outlined earlier, not all VNFs are shareable, $S(v)$ is a flag to determine if a VNF v

is shareable. The time a VNF becomes part of sfc_j, that VNF's inflow $F_{in}(v_i^j)$ and outflow $F_{out}(v_i^j)$ are determined. Due to operation dynamics and the fact that some VNFs, like firewalls and optimizers, may drop or compress inflow, both inflow and outflow are subject to change $(F_{out}(v_i^j) \leq F_{in}(v_i^j))$. The substrate network of nodes that hosts VNFs is represented as a graph $G(N, E)$, where N is the set of nodes and E is the set of links. Each node $n \in N$ has compute resources, $CPU\ cores$ and $memory$, and each link $e \in E$ has $bandwidth$ capacity bw_c and propagation delay Del. Table I lists detailed description of substrate network and SFC parameters.

TABLE I. System Parameters Description

Parameter	Description
$cpu_{c\|av}(n)$	CPU capacity | available in cores of node $n \in N$
$ram_{c\|av}(n)$	RAM capacity | available in GBs of node $n \in N$
$e_{nn'}$	A link exists from node n to node n', $n, n' \in N$
$bw_{c\|av}(e_{nn'})$	BW capacity | available in Mbps of link $e_{nn'}$
$Del(e_{nn'})$	Propagation delay of link $e_{nn'}$
$cpu(v_i)$	CPU cores required for VNF $v_i \in V$
$ram(v_i)$	RAM GBs required for VNF $v_i \in V$
$F_{max}(v_i)$	Maximum inflow VNF v_i can handle
$S(v_i)$	Flag to indicate VNF v_i is shareable
sfc_j	SFC request j
$\|sfc_j\|$	Number of VNFs in sfc_j
v_i^j	The i^{th} VNF of sfc_j
$F_{in}(v_i^j)$	Actual inflow that VNF v_i will be serving
$F_{out}(v_i^j)$	Outflow VNF v will produce
$Del(sfc_j)$	Maximum end-to-end delay of sfc_j

B. Problem Formulation

Algorithm 1 describes the proposed PSVS scheme. The scheme hinges on the placement algorithm and the required resources prediction component. The placement algorithm is modeled as an integer program with binary decision variables and utilizes *VNF sharing*. PSVS utilizes a discretized time scale of TSs. First, at the beginning of each TS, PSVS terminates and releases utilized resources of running $sfc_{pr|be} \in Run_{pr|be}$ whose time-to-live (TTL) is zero and moves them to $Com_{pr|be}$ list. If SFC's TTL is not zero, PSVS decreases the TTL by one. Second, to prioritize sfc_{pr}, PSVS attempts to satisfy received requests in Rec_{pr} queue first. Finally, if there are requests in Pen_{be} queue, the predicted required resources in the next ω TSs are utilized to either satisfy some or all of the requests in Pen_{be} or defer deployment to leave room for future sfc_{pr} requests to arrive. We utilize no Pen_{pr} queue because sfc_{pr} requests gets rejected if not immediately satisfied. Similarly, no Rej_{be} list is used, since sfc_{be} requests can tolerate waiting for deployment.

Algorithm 1: PSVS

```
// SimDur: Simulation duration in time slots
   (TSs), $\omega \in [1-3]$ TSs Lookahead window
   length, and $\alpha \in [1-3]$ Resource
   safety-margin (a scaling factor)
Input  : net-Model, SimDur, $\omega$, $\alpha$
Init.  : $Rec_{pr}$, $Run_{pr|be}$, $Pen_{be}$, $Rej_{pr}$, $Com_{pr|be}$
Output: Different queues\lists and collected
        statistics

1  for $i \leftarrow 1$ to SimDur do
2      $Rec_{pr} \leftarrow$ received $sfc_{pr}$ requests
3      $Pen_{be} \leftarrow$ received $sfc_{be}$ requests
       // Update TTL of running SFCs
4      foreach $sfc_{pr|be}$ in $Run_{pr|be}$ do
5          if ttl($sfc_{pr|be}$) $= 0$ then
               // Releases resources utilised
                  by finished sfc
6              $Com_{pr|be} \leftarrow sfc_{pr|be}$
7          else decTTL($sfc_{pr|be}$)
8      foreach $sfc_{pr}$ in $Rec_{pr}$ do
           // Using IQCP & Gurobi solver
9          sol $\leftarrow$satisfy($sfc_{pr}$,net-Model)
10         if sol $\neq \emptyset$ then
11             deploy($sfc_{pr}$,net-Model)
12             $Run_{pr} \leftarrow sfc_{pr}$
13         else // $sfc_{pr}$ can not tolerate
              waiting for deployment
14             $Rej_{pr} \leftarrow sfc_{pr}$
       // Required resources/$sfc_{pr}$ requests
          expected to arrive in $\omega$ TSs
15     $ReqrdRes_{pr} \leftarrow$
        predReqResources($\omega$)
       // $AvailRes$: free resource in current
          TS and in next $\omega$ TSs
16     $AvailRes \leftarrow$ getAvailResources($\omega$)
17     $ExtraRes_{be} \leftarrow 0$
18     if $AvailRes > (\alpha * ReqrdRes_{pr})$ then
19         $ExtraRes_{be} \leftarrow AvailRes - (\alpha$
            $* ReqrdRes_{pr})$
20     while $(\exists sfc_{be}$ in $Pen_{be})$&$(ExtraRes_{be}$
        $\neq 0)$ do
21         sol $\leftarrow$satisfy($sfc_{be}$,net-Model)
22         if sol $\neq \emptyset$ then
23             deploy($sfc_{be}$,net-Model)
24             $Run_{be} \leftarrow sfc_{be}$
25             $ExtraRes_{be}$
                $\leftarrow ExtraRes_{be}$-usedRes($sfc_{be}$)
           // else sfc stays in $Pen_{be}$
```

*1) **Placement Algorithm**:* With an SFC request sfc_j consisting of VNFs $v_i, i \in [1 - |sfc_j|]$, the decision variables are: χ_{in}^j: if a new instance of VNF v_i belonging to sfc_j is to be placed at node n; and Υ_{in}^j: if VNF v_i of sfc_j is to share and become the guest of a deployed underutilized VNF of the same type at node n. Queues, decision variables, and parameter descriptions are in Table II.

*a) **Objective Function**:* The objective function, equation (1), is designed to favour the placements that minimizes the overall cost and resource utilization by preferring sharing underutilized VNFs over instantiating new ones. The cost of utilizing a deployed shareable VNF in only the bandwidth cost, compared to instantiating a new VNF instance which includes cpu and ram in addition to bandwidth cost. The feasible solution has to satisfy constraints 2-9.

$$min \sum_{i=1}^{|sfc_j|} \sum_{n \in N} [cpu(v_i^j) U_{cpu}^c(n) + ram(v_i^j) U_{ram}^c(n)] \chi_{in}^j + F_{out}(v_i^j) U_c bw[\chi_{in}^j + \Upsilon_{in}^j] \tag{1}$$

TABLE II. Queues, Decision Variables and Constants

Variable/Queue	Description
χ_{in}^j	For placing a new VNF v_i of sfc_j at node n
Υ_{in}^j	For sharing the flow of VNF v_i of sfc_j with already deployed VNF of same type at node n
D_n^i	VNF of same type as v_i is deployed at node n
$F_{av}(v_n^i)$	Available unused flow of v_i at node n
$U_{cpu}^c(n)$	Unit cost of cpu at node n
$U_{ram}^c(n)$	Unit cost of ram at node n
$U_c(bw)$	Unit cost of bw at all links
Rec_{pr}	Queue of new Pr SFCs until deployed
Pen_{be}	Queue of new BE SFCs until deployed
Rej_{pr}	List of rejected Pr SFCs
$Run_{pr\|be}$	List of deployed Pr or BE SFCs
$Com_{pr\|be}$	A list of finished Pr or BE SFCs

$$\sum_{n \in N} \chi_{in}^j + \Upsilon_{in}^j = 1 \quad , \quad \forall i \in [1, |sfc_j|] \tag{2}$$

$$\sum_{n \in N} \chi_{in}^j + D_n^i \, S(v_i) \, \Upsilon_{in}^j = 1, \quad \forall i \in [1, |sfc_j|] \tag{3}$$

$$F_{in}(v_i^j) \, \Upsilon_{in}^j \le F_{av}(D_n^i), \ \forall n \in N \ \& \ \forall i \in [1, |sfc_j|] \tag{4}$$

$$\sum_{i=1}^{|sfc_j|} cpu(v_i^j) \, \chi_{in}^j \le cpu_{av}(n), \quad \forall n \in N \tag{5}$$

$$\sum_{i=1}^{|sfc_j|} ram(v_i^j) \, \chi_{in}^j \le ram_{av}(n), \quad \forall n \in N \tag{6}$$

$$e_{nn'} \, (\chi_{in}^j + \Upsilon_{in}^j)(\chi_{(i+1)n'}^j + \Upsilon_{(i+1)n'}^j) = 1 \quad \forall n, n' \in N \ \& \ \forall i \in [1, (|sfc_j| - 1)] \tag{7}$$

$$\sum_{n \in N} \sum_{n' \in N} F_{out}(v_i^j)(\chi_{in}^j + \Upsilon_{in}^j)(\chi_{(i+1)n'}^j + \Upsilon_{(i+1)n'}^j) \le bw_{av}(e_{nn'}), \quad \forall i \in [1, (|sfc_j| - 1)] \tag{8}$$

$$\sum_{i=1}^{|sfc_j|-1} \sum_{n \in N} \sum_{n' \in N} Del(e_{nn'}) \, (\chi_{in}^j + \Upsilon_{in}^j) (\chi_{(i+1)n'}^j + \Upsilon_{(i+1)n'}^j) \le Del(sfc_j) \tag{9}$$

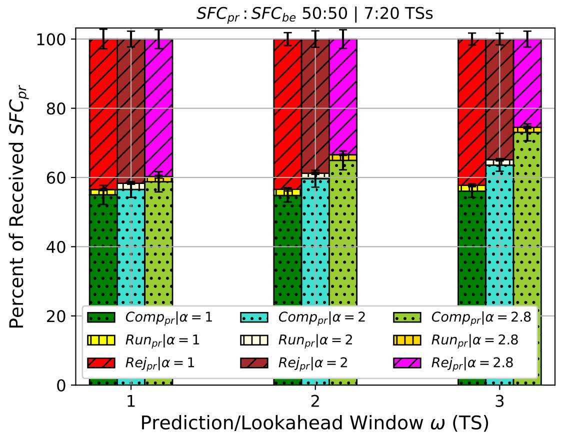
*2) **Required Resources Prediction**:* The assumption here is that a simple prediction model exists and based on the historical data, it can predicts number of sfc_{pr} requests to arrive in the next ω TSs and their required resources. As can be seen in line 18 of Algorithm 1, when comparing predicted required resources ($ReqrdRes_{pr}$) to the available resources ($AvailRes$), we utilize a *safety-margin* α.

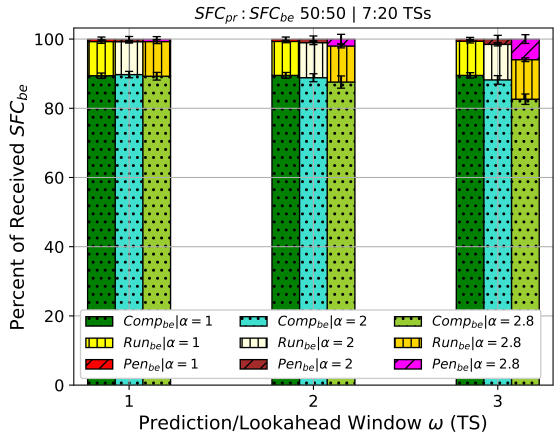
The main purpose of using α is to compensate for the difference in TSs between the size of lookahead window $\omega \in [1, 3]$ and the duration of sfc_{be} requests (typically 10 or 20 TSs). For example, for $\alpha = 1$, if the $AvailRes$ is greater than predicted $ReqrdRes_{pr}$, a deployed sfc_{be} in the current TS will have a lasting impact on resources availability that extends beyond the lookahead window, hence the sfc_{pr} requests rejections. The value of *safety-margin* α should be carefully empirically selected/tuned. On the one hand, small values of α will lead to higher rejection rate. On the other hand, large values cause starvation for the pending sfc_{be} requests as it will be over restrictive and seize more resources for future sfc_{pr} requests, resulting in lower system utilization and prolonged waiting times for sfc_{be} requests. It worth noting that, the $AvailRes$ are those resources free in the current TS in addition to resources that will be released in the next ω TSs, as running SFCs are finished (TTL=0) and moved to $Com_{pr|be}$ list. Extra resources ($ExtraRes_{be}$) are those available resources that exceed the α-scaled $AvailRes$ and are utilized to satisfy one or more requests pending in the Pen_{be} queue (see Algorithm 1 lines 19-25).

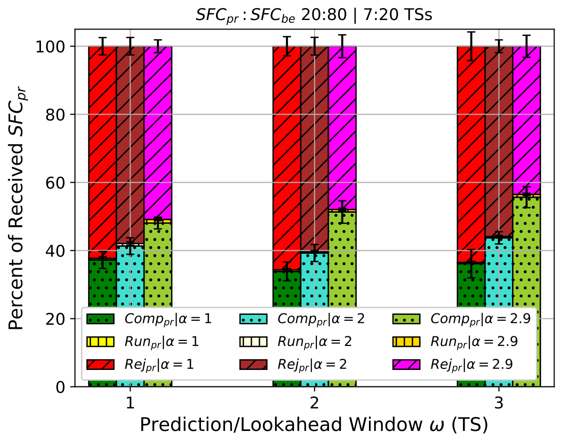
IV. Performance Evaluation

A. Simulation Framework

We developed a Java-based simulation environment to synthetically generate SFC requests and the network model (NSFNET network model that has 13 nodes and 32 directional links). The SFC requests arrival rate per TS follows a Poisson distribution with average rate $\lambda = 2$. SFC length, number of VNFs, is drawn from a uniform distribution $|sfc_j| \sim U[4, 7]$ [23]. The service time, i.e., SFC duration in TSs, is fixed, where $sfc_{pr} = 7$


(a) Moderately-loaded


(b) Moderately-loaded


(c) Highly-loaded

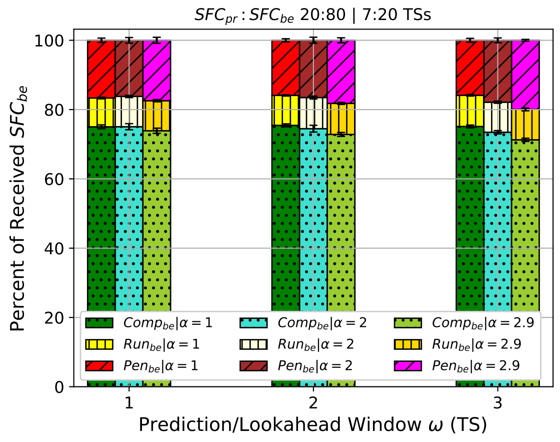

(d) Highly-loaded

Fig. 5. Rejected/Pending, running, and completed (%) of received sfc_{pr} and sfc_{be} requests for two system configurations (a)&(b) moderately-loaded, and (c)&(d) highly-loaded.

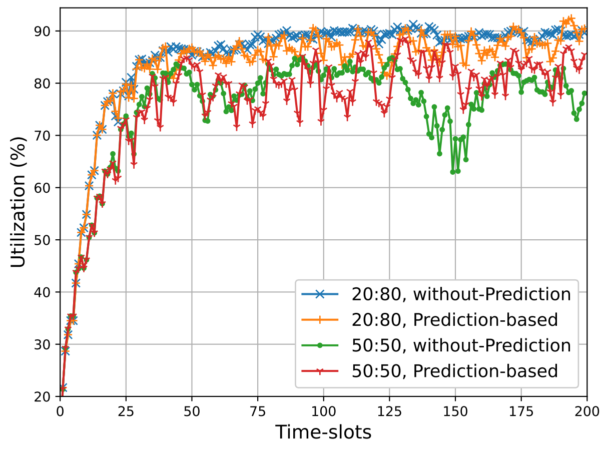


Fig. 6. utilization throughout simulation time (200 TSs).

and $sfc_{be} \in \{10, 20\}$. The ratio of sfc_{pr}:sfc_{be} requests is either *'50:50'* or *'20:80,'* and the order of arrival is shuffled. Each experiment is repeated ten times where the network model, nodes, and links' resources are the only constant. Across the ten repetitions, the variations are: number and type of requests per TS, the length and type of VNFs in sfc requests, and sfc requests QoS/performance requirements (end-to-end delay). We utilized a list of on-boarded VNFs that contains 16 VNFs, 60% of which are shareable. The placement scheme is solved using the Gurobi solver [24].

B. Numerical Results and Analysis

To assess the impact of both lookahead window size ω and *safety-margin* α on the $Rej_{pr}\%$, we experimented with different values of ω and α, and with different system loads. As can be seen in Figures 5a and 5b, the *moderately-loaded* system witnesses an increase in completed requests ($Com_{pr|be}$), significantly fewer rejected requests (Rej_{pr}), and about the same percentage of received requests that are in the running state ($Run_{pr|be}$). When comparing the rejection rate of $\alpha = \{1, 2, 2.8, 2.9\}$ and $\omega = 3$, we can conclude that the *safety-margin* has a significant impact on reducing the sfc_{pr} rejection rate. The same effect is carried over to the *highly-loaded* system, as shown in Figures 5c and 5d. Even though, the $+50\%$ reduction in the rejection rate of sfc_{pr} requests, the Pen_{be} queue did not expand that much (compare the size of Pen_{be} of $\omega = 3$ and $\alpha = 2.9$ for the *highly-loaded* system.

The best values that α converged to might seem a magic number. However, we can easily find simpler more deterministic systems that use the same empirical methodology to determine the best value/range for a probability/factor that maximizes an important *KPI*. For example, how the *'p'* in *p*-persistent carrier sense multiple access (CSMA) is calculated to maximize throughput [25].

The utilization of both *moderately* and *highly-loaded* systems, shown in Figure 6, reveals that PSVS does not increase the completion and reduces the rejection rates of sfc_{pr} requests by totally ignoring the sfc_{be} requests and leaving the system resources idle. Furthermore, PSVS enhances the overall utilization of the *moderately-loaded* system. For a detailed system utilization, we can consult Figure 5 and see that the size of $Run_{pr|be}$ of both system loads is almost the same for different values of ω and α. There are two faces to the PSVS schemes,

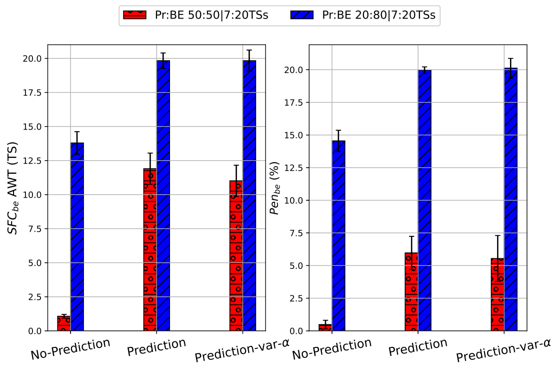
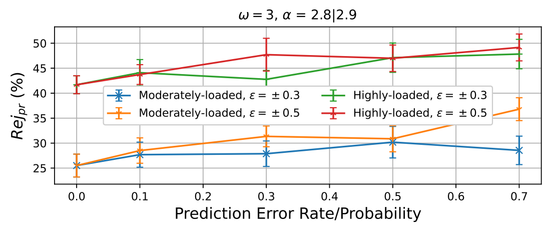
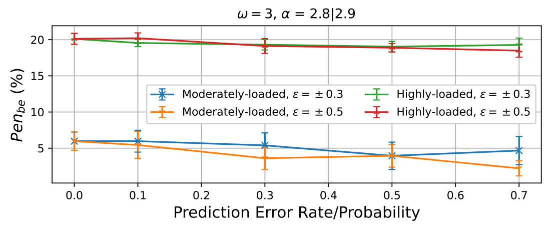


Fig. 7. Left: sfc_{be} average waiting time (AWT). Right: percentage of pending sfc_{be} requests. *Prediction* is using $\alpha = 2.8$ for 50:50 systems and $\alpha = 2.9$ for 20:80 system. The var-α uses a varying *safety-margin* with lower-bound=2.8, upper-bound=2.9, and the step=+0.1,-0.01.

the impact on rejection rate of sfc_{pr} requests and on the AWT and pending sfc_{be} requests. Both the AWT and percentage of pending sfc_{be} requests are reported in Figure 7. PSVS increased the average waiting time (AWT) and the percentage of pending sfc_{be} requests by 43% and 38%, respectively. In addition to compensating the difference between the short lookahead window (for accurate



(a)



(b)

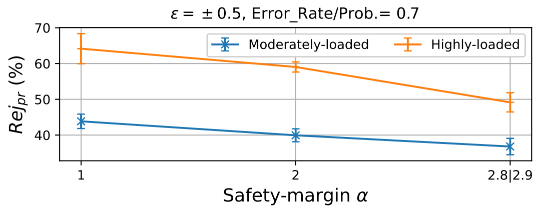
Fig. 8. The impact of prediction error value and rates/probabilities on (a) Rejected Pr SFCs requests and (b) Pending BE SFC requests, for moderately- and highly-loaded systems.

predictions) and the 20TS-long sfc_{be} requests, the *safety-margin* α helps address the uncertainty arising from the VNF sharing (the used resources to satisfy requests varies depending on the current snapshot of deployed SFCs) and increases the robustness of PSVS and its resilience to prediction errors. To evaluate the robustness of PSVS results and demonstrate the importance of *safety-margin* α, we experimented with error rates/probabilities up to 70% of predicted required resources and with prediction error value ε up to $\pm 50\%$. As shown in Figure 8a, the biggest increase in the *highly-loaded* system's rejection rate is only 17% in face of a the extreme error values and rates. As shown in Figure 8b, the Pen_{be} queue size did not grow.

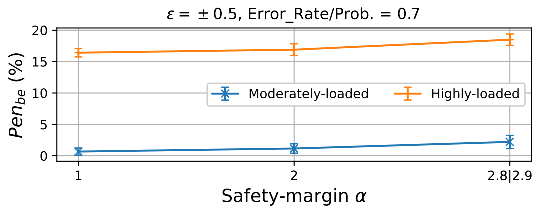
Figure 9 shows the impact of *safety-margin* α on sfc_{pr} requests rejection rate under extreme prediction error values and rates/probabilities. For both system loads, the higher the value of α, the less the rejection rate. In the vein, the increase in the Pen_{be} queue size is not concerning, see Figure 9b.

V. Conclusions and Future Work

This paper proposed PSVS to reduce the rejection rate when provisioning time-sensitive Pr services at the edge. PSVS utilizes both the predicted required resources and a *safety-margin* to address the difference between ω and sfc_{be} requests duration and to provide resiliency against prediction errors. We experimented and evaluated different lookahead window sizes ω and *safety-margin* α and concluded the best values to balance the reduction in sfc_{pr} requests rejection rate and sfc_{be} requests AWT. Finally, more reduction in the rejection rate is attainable; however, the sfc_{be} requests will suffer more starvation.



(a)



(b)

Fig. 9. The impact of *safety-margin* on sfc_{pr} requests rejection rate under extreme prediction error values and rates/probabilities, (a) Rejected Pr SFCs requests and (b) Pending BE SFC requests.

This might be desirable in emergencies where time-critical premium services must be immediately satisfied.

Rejections of sfc_{pr} requests are unavoidable with prediction window size ω smaller than sfc_{be} requests average duration. However, to get accurate predictions, we had to continue using small lookahead window sizes. However, in an environment where lower-priority services can be suspended and the pre-emption cost is bearable, a zero sfc_{pr} rejections might be achievable by suspending one or more running BE services and pre-empting resources to premium services. Future research includes, a pre-emption criterion that pre-empts resources for sfc_{pr} requests and minimize the disturbance to BE services.

References

[1] A. Ghosh, A. Maeder, M. Baker, and D. Chandramouli, "5g evolution: A view on 5g cellular technology beyond 3gpp release 15," *IEEE access*, vol. 7, pp. 127 639–127 651, 2019.

[2] O-RAN Alliance, "O-RAN Architecture Description v03.00," Tech. Rep., 2021.

[3] N. Kazemifard and V. Shah-Mansouri, "Minimum delay function placement and resource allocation for open ran (o-ran) 5g networks," *Computer Networks*, vol. 188, p. 107809, 2021.

[4] Ericsson, "Time-Critical Communication: Leading the next wave of 5G innovation," Ericsson, Technical-Overview, 2021. [Online]. Available: https://www.ericsson.com/4a9e9f/assets/local/internet-of-things/docs/19102021-time-critical-communication-brochure.pdf

[5] C. Grasso, K. E. KN, P. Nagaradjane, M. Ramesh, and G. Schembra, "Designing the tactile support engine to assist time-critical applications at the edge of a 5g network," *Computer Communications*, vol. 166, pp. 226–233, 2021.

[6] C. Liu, K. Liu, S. Guo, R. Xie, V. C. S. Lee, and S. H. Son, "Adaptive offloading for time-critical tasks in heterogeneous internet of vehicles," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7999–8011, 2020.

[7] A. Jain and D. S. Jat, "An edge computing paradigm for time-sensitive applications," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 798–803.

[8] MarketsandMarkets, "Edge Computing Market by Component, Application, Organization Size, Vertical and Region - Global Forecast to 2025," https://www.marketsandmarkets.com/Market-Reports/edge-computing-market-133384090.html, accessed: 16-03-2022.

[9] ETSI, "Network function virtualization: An introduction, benefits, enablers, challenges & call for action," in *SDN and OpenFlow World Congress*, Oct. 2012, pp. 1–16.

[10] H. Yu, Z. Zheng, J. Shen, C. Miao, C. Sun, H. Hu, J. Bi, J. Wu, and J. Wang, "Octans: Optimal placement of service function chains in many-core systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2202–2215, 2021.

[11] M. Li, Q. Zhang, and F. Liu, "Finedge: A dynamic cost-efficient edge resource management platform for nfv network," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–10.

[12] C. Mei, J. Liu, J. Li, L. Zhang, and M. Shao, "5g network slices embedding with sharable virtual network functions," *Journal of Communications and Networks*, vol. 22, no. 5, pp. 415–427, 2020.

[13] L. M. Contreras, A. Solano, F. Cano, and J. Folgueira, "Efficiency gains due to network function sharing in cdn-as-a-service slicing scenarios," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 348–356.

[14] B. Yi, X. Wang, and M. Huang, "A generalized vnf sharing approach for service scheduling," *IEEE Communications Letters*, vol. 22, no. 1, pp. 73–76, 2017.

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

[16] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing service deployment cost through vnf sharing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2363–2376, Dec 2019.

[17] N. Siasi and A. Jaesim, "Priority-aware sfc provisioning in fog computing," in *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, 2020, pp. 1–6.

[18] S. Park, H.-G. Kim, J. Hong, S. Lange, J.-H. Yoo, and J. W.-K. Hong, "Machine learning-based optimal vnf deployment," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 67–72.

[19] S. Lange, H.-G. Kim, S.-Y. Jeong, H. Choi, J.-H. Yoo, and J. W.-K. Hong, "Machine learning-based prediction of vnf deployment decisions in dynamic networks," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–6.

[20] S. Pandey, J. W.-K. Hong, and J.-H. Yoo, "Gru and edgeq-learning based traffic prediction and scaling of sfc," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 124–132.

[21] H.-G. Kim, D.-Y. Lee, S.-Y. Jeong, H. Choi, J.-H. Yoo, and J. W.-K. Hong, "Machine learning-based method for prediction of virtual network function resource demands," in *2019 IEEE conference on network softwarization (NetSoft)*, 2019, pp. 405–413.

[22] A. Mohamad and H. S. Hassanein, "Psvshare: A priority-based sfc placement with vnf sharing," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 25–30.

[23] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks," Internet Engineering Task Force, Internet-Draft, Jan. 2019, work in Progress.

[24] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com

[25] H. Takagi and L. Kleinrock, "Throughput analysis for persistent csma systems," *IEEE transactions on communications*, vol. 33, no. 7, pp. 627–638, 1985.