Dynamic Network Function Instance Scaling Based on Traffic Forecasting and VNF Placement in Operator Data Centers

Hong Tang, Danny Zhou and Duan Chen

Abstract—Traffic in operator networks is time varying. Conventional network functions implemented by black-boxes should satisfy the peak traffic requirement, and hence result in low resource utilization. Thanks to the emergence of Virtual Network Function (VNF), which is realized by running networking software on Virtual Machines (VMs), the operator can dynamically scale in or scale out the VNF instances and hence save the required resources. In this paper, we introduce how the dynamic VNF scaling is implemented in practical operator Data Center Networks (DCNs). Firstly, we analyze the traffic characteristics in our operator networks, and introduce how the VNFs are organized in a common operator DCN. Based on these backgrounds, we not only propose a traffic forecasting method, but also design two VNF placement algorithms to guide the dynamic VNF instance scaling. Through both the implementation in a real operator network and extensive real trace driven simulations, we demonstrate that our dynamic VNF instance scaling system can achieve higher service availability and save the VNF resources (e.g. CPU and memory) by up to 30%.

Index Terms—Data Center Networks (DCNs); Virtual Network Functions (VNFs); Dynamic Service Function Chaining (SFC).

1 Introduction

In operator networks, the traffic should pass through multiple middleboxes that provide different network functions. Traditionally, these middleboxes are implemented as closed black-boxes, where both hardware and software are tightly coupled as they are often supplied or integrated by the same vendor [1]–[3]. To decouple the software from the hardware, Network Function Virtualisation (NFV) is proposed to implement middleboxes in software running on VMs [4]–[6]. Such a software based middlebox is called a Virtual Network Function (VNF); its dataplane can be realized by multiple VMs, each is called a VNF instance.

Conventionally, traffic should be steered to traverse middleboxes implementing the required VNFs in a specific order satisfying business requirement. For example, if the access to a server needs to be highly restricted, its incoming traffic may first traverse a firewall (FW) and then an Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS). We call such a ordered set of middleboxes as a Service Function Chain (SFC). In this paper, we focus on enforcing SFC with VNF technology.

With VNF technology, a great benefit is that the operator can dynamically scale in VNF instances to fulfill the SFC requirement when there is a increasing amount of traffic, while scale out some of them to save operation cost, such as electricity, when there is no heavy traffic (and hence less SFC services requirement) in the network. There are a lot of existing works on how to place the VNF instances and steer traffic in the network to fulfill the SFC requirement, such as E2 [7], Stratos [8] and [9]. However, most of these works assume the VNF instances can be instantiated without any limitations in the Data Centers (DCs), which is not the case in reality. For the management reasons, the network operators usually adopt two types of SFC models: SFC Run-to-Complete in a Rack [10, 11] and Cross Rack Pipelined SFC [12]–[14]. In the first model, each SFC must be completed under one rack, while in the second model, each rack only hosts instances for the same type of VNF, and each SFC should be completed by steering the traffic across multiple racks.

When either the SFC Run-to-Complete in a Rack or Cross Rack Pipelined SFC models is adopted, the existing VNF instance scaling methods do not work any more. For example, if a VNF provides services to multiple SFCs, when this VNF is heavily loaded and we need to set up a new VNF, it is difficult for us to determine where to place this new VNF instance as we do not know from which SFC the increasing traffic comes. Consequentially, we do not know how to update the traffic routing. To solve this problem, we need to forecast the traffic amount and then overall plan the VNF placement in order to minimize the required sources to serve all the traffic. Different from previous traffic forecasting works, we cannot underestimate the traffic amount which may result in a low service availability, while we cannot overestimate the traffic amount too much which leads to resource waste. Therefore, how to estimate the traffic upper bound but minimize the overestimation is a key challenge of our work.

Based on the traffic estimation, we need to determine the VNF

• Hong Tang is with School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, and is also with Guangzhou Institute of China Telecom. Email: tangh7@mail2.sysu.edu.cn
• Danny Zhou is with Huawei Technologies Corp.. Email: danny.zhou@huawei.com
• Duan Chen is with Nokia Shanghai Bell Co., Ltd.. Email: duan.chen@nokia-sbell.com

placement scheme such that the required amount of resources can be minimized. Unfortunately, regardless of which SFC deployment model, i.e. Run-to-Complete model or Cross Rack Pipeline model, is adopted, the VNF placement problem is NP-hard and is difficult to solve in a timely manner. Accordingly, how to solve the VNF placement problem efficiently is another challenge. With the VNF placement scheme calculated based on the forecasted traffic amount, we can proactively scale in VNF instances to accomodate the traffic increases, and scale out the VMs if there is a decreasing amount of traffic. As far as we know, this is the first work that reports how to realize the dynamic SFC provisioning in the ICT industry.

The main technical contributions of this paper can be summarized as follows:

- A traffic forecasting method to enable dynamic VNF instance scale in and scale out
- Efficient algorithms to solve how to provision the VNF instances in the operator networks for both of the Run-to-Complete model and the Cross Rack Pipelined SFC model
- Implementation of dynamic SFC provisioning system
- Large scale real data driven simulation to show the system efficiency

The rest of this paper is structured as follows. Section 2 briefly reviews the related works of this paper, and Section 3 presents the background of our work. Then, the traffic forecasting problem and VNF placement problems are formulated in Section 4. In Section 5 and 6, we discuss how to solve the traffic forecasting problem and VNF placement problem in detail, respectively. In Section 7, we discuss some practical issues and the implementation of the the system we propose in this paper. Large scale real trace driven simulations are performed in Section 8. We conclude this paper in Section 9.

2 Related Work

Network Function Virtualisation (NFV) addresses the issues of tight coupling of hardware and software and vendor lock-in for middleboxes, by implementing network functions purely in VMs running on off-the-shelf commodity servers. Recently, there are some researches in network function placement [7, 8, 15]–[18]. CoMb [15] exploits the opportunities of consolidating software middleboxes to save cost, but it requires that the entire SFC is realized in a single thread. MIDAS [16] removes this constraint and assigns required Network Functions (NFs) to multiple CoMb servers. Both CoMb and MIDAS are thread-based scheme to realize SFC. However, the thread-based scheme does not satisfy the performance isolation property required in multi-tenant clouds.

For performance isolation purpose, many researchers are focusing on the VM based VNF placement scheme. E2 [7] realizes each VNF as a VM and consolidates the VMs of an SFC to a server, and hence reduces the inter-server traffic. This scheme inherently assumes the run-to-complete model without considering the pipeline model. In addition, it only proposes a myopic scheme which cannot optimize the SFC placement in a global view, and hence only derives a suboptimal solution. [19] and [20] present algorithms to minimize the amount of required computing resource. However, both of them do not consider any model adopted in the real operator networks and hence are difficult to deploy in operator networks. VM placement (VMPP) places individual VMs to physical hosts, such that the cross-rack traffic can be minimized [21]. It does not consider the SFC requirement. Though we can extend its algorithm to take the SFC requirement into consideration, it suffers from the same problem that neither the run-to-complete model nor the pipeline model are suitable. VNP-OP [17] studies how to jointly optimize the VM placement and traffic routing to minimize the deploy cost and forwarding cost. PACE [18] also focuses on the VNF placement to accommodate as many requests as possible, but it assumes that all the NF requirements in an SFC are unordered.

Another important part of our work is traffic forecasting. There are also a lot of works on this area [22]–[26]. [22] leverages neural network to forecast the traffic on an hourly basis, and while [24] focuses on the large time-scale traffic variation forecasting, which are both too coarse-grained for our work. [23] introduces the cluster technology into the conventional time series models to improve the traffic forecasting. [25] and [26] both adoptively learn the traffic trend and leverage the forecasting results to guide the VNF provisioning. However, neither of them focus on the upper bound of the traffic amount, and hence, they cannot ensure the high service availability as in our work.

3 Background

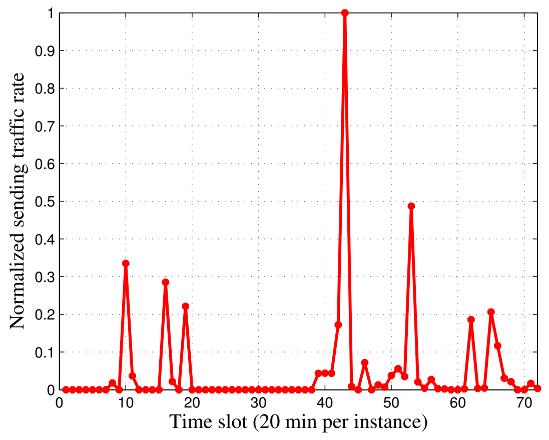
In this section, we present some backgrounds about our work. At first, we show some real data analyses that motivate our work in Section 3.1. Then, we investigate the time for setting up or migrating a VM for VNF through experiments in real testbed in Section 3.2. At last, we introduce the physical architectures, i.e. SFC models, that are leveraged to deploy VNFs in Section 3.3.

3.1 Traffic Analysis

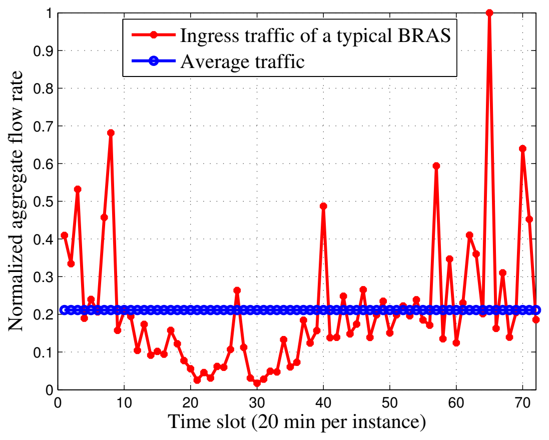
In operator networks, the traffic injected into the network is continuously varying. Fig. 1 shows the traffic rate that a typical subscriber sends into the network, and the ingress traffic rate of the BRAS router hosting this typical subscriber, which are collected from our operator network. This figure shows the traffic variance in 2 days by sampling the traffic rate every 20 minutes. For confidentiality issue, we normalize the traffic rate onto [0,1].

From 1(a), we can see that the traffic transmission rate of a subscriber centers around several time instances, and for most of the time, there is merely little traffic. This is due to the fact that the traffic rate is determined by the subscriber behavior and the application running on the end devices of this subscriber. Usually, a subscriber is only active in a few time instances, which is unpredictable. Ideally, if there are plenty of subscribers in an area and their network utilization evenly distributed among the time axis, we can observe a stable aggregate traffic rate at the aggregate router. Accordingly, we also collect the ingress traffic rate of this typical subscriber from the BRAS router.

Fig. 1(b) shows the realistic traffic rate. Unfortunately, we do not observe stable traffic rate, and the ingress traffic rate is even more unstable than that of a specific subscriber. For most of the time, the ingress traffic rate is relatively low, and the peak ingress traffic rate is about 5 times of the average ingress traffic rate. More importantly, there are only 22 out of 72 time instances in which the ingress traffic rate is larger than the average rate. Therefore,



(a) Traffic rate of a typical subscriber.



(b) Ingress traffic rate of corresponding BRAS

Fig. 1. Traffic characteristic analysis.

if we always reserve enough VNF resources to accommodate the peak traffic, there must be lots of resource waste. Based on this observation, we decide to *dynamically scale in more VMs to adopt the traffic increase and scale out VMs to save resources when there is relatively little traffic in the network*. In this way, we can not only ensure the network service availability, but also save physical resources and enhance the resource utilization.

3.2 Time Consumption of Operations

To dynamically scale in and scale out VMs in the data center, we also need to investigate the time consumption of different VM operations. Such investigation is based on our private testbed (configuration details are presented in Section 7.2), which is with the same configuration in the real operator DCNs. In addition, we also investigate the time to migrate flows to different paths. These investigations are summarized in Tab. 1.

TABLE 1
Time consumption of VM operations

Operation	Time Consumption
Set up new VM	6 minutes
VM migration	3 minutes
VM deletion	5 seconds
Flow migration	2 seconds

From the table above, we can see that it is time consuming to scale in a new VM. As shown in Tab. 1, it needs about 6 minutes to scale in a new VM. Hereby, we cannot dynamically scale in a VM when the traffic rate has already increased. In this case, we should cache the traffic for a long time (if there is enough cache capacity), or we should reject some of the traffic as there are not enough resources to provide services to it.

In addition, migrating a VM needs about 3 minutes, which is less than that for setting up a new VM. However, we need to try to limit the use of VM migration for the purpose of saving resources. This is due to three reasons: 1) 3 minutes is still a long time for a request or a flow stream; 2) migrating one VM to another requires keeping both VMs on, and hence it does not really reduce the physical resource consumption; 3) VM migration results in more traffic to be injected into the network. However, it is still a good way to optimize the traffic in the network. Accordingly, we do not add any limitation to the VM migration when we optimize our system. To prevent the abuse of VM migration, we discuss how to limit the VM migration in real system in Section 7.1.

Furthermore, VM deletion spends only 5 seconds. Hereby, whenever there is little traffic in the network and resources are underutilized, we can scale out some of the VMs in order to decrease the physical resource consumption.

At last, we can see that migrating a flow to a different path requires very little time. For most of the online services, an occasional 2-second delay may not greatly impact the customer experiences. Based on this fact, when we optimize the network resource utilization, we can migrate flows to consolidate them on fewer VMs. With this strategy, we can empty more VMs and scale them out to save physical resources.

3.3 Architecture to Deploy Service Function Chains

To deploy SFCs in an operator data center, we have four typical SFC deployment models as shown in Fig. 2. Fig. 2(a) and 2(b) show the VNF relationship in the physical server level. One deploy method, as shown in Fig. 2(a), is that each physical server sets up multiple VMs and complete the entire SFC by a single physical server, i.e. consolidate the SFC in a single server. Such a method is deeply discussed in [15]. The second method, as shown in Fig. 2(b), is to set up a single type of VNF on each physical server. In this case, to fulfill an SFC, the traffic should traverse multiple physical servers.

Fig. 2(c) and Fig. 2(d) are how we complete a SFC in the rack level. In Fig. 2(c), each SFC must be completed in the same rack. In this case, the deployment models in both Fig. 2(a) and 2(b) can be adopted. While in Fig. 2(d), all the physical servers under the same rack can only host the same type of VNF instances, and consequentially, only the cross server pipeline SFC deployment model shown in Fig. 2(b) can be adopted in this case.

In each rack, all the servers are connected by a top-of-rack (ToR) switch. For the cross rack connection, there are two ways: 1) all the racks are connected by a nonblocking fabric, such as Fattree [27] and VL2 [28]; 2) all the racks are linearly connected, i.e. any traffic should traverse through all the ToR switches from its starting rack to the ending rack on the pipeline. It should be noted that, when all the racks are linearly connected, the traffic can only delivered in one direction, i.e. if there is traffic from rack r to rack $r+1$, in order to simplify the management.

It is worth noting that, theoretically, the SFC deployment models discussed above add some constraints into the deployment and may degrade the network performance or reduce the resource utilization, such as increasing the cross-server (or cross-rack) traffic. However, in operator's perspective, such constraints are necessary for the management purpose. To adopt the run-to-complete model, the operator can eliminate all the cross-rack (or cross-server) traffic. Furthermore, when some SFC fails, the operator can quickly allocate the failed rack/server. The cross-rack pipelined SFC model is also proposed for the management issue. On the one hand, usually, the operator maintains a large group of staffs for each type of VNF. Host only one type of VNF under a certain rack can greatly simplify the management cost and reduce the risk of misoperation. On the other hand, it is also good for the trouble shooting to find out which VNF suffers from unexpected problems.

In operator data centers, there are very few type of VMs that are used to host VNFs. For simplicity, we assume all the VMs are the same in the network. However, it does not mean all the VMs can serve the same amout of traffic. When a VM realize different types of VNFs, the traffic amount can be served may vary.

4 Problem Statement

To provide services with high availability, at least, we need to do two things. First, we should forecast the traffic amount at each time instance. This is because that the traffic in the network is time varying. We need to pre-calculate the VNF placement and traffic steering scheme for the traffic arriving a few minutes later. In this case, we have enough time to scale in new VMs to serve the traffic for the next time instance; second, with this traffic forecasting, a VNF placement scheme should be proposed to serve all the possible traffic with minimal amount of resources. In this section, we first formulate the problems to be solved in our work, and will design algorithms to solve the proposed algorithms in the next section.

For clear presentation, we summarize the notations that will be used in this paper in Tab. 4.

4.1 Formulation of traffic forecasting

To provide the high service availability, we should always ensure there are enough resources to serve all the traffic in the system. Accordingly, the objective of the traffic forecasting is the *upper bound* of the traffic amount, rather than the exact traffic amount as previous works [22]–[26]. Say the traffic amount at time instance k is $f[k]$, the traffic forecasting of time instance T can be formulated as

$$B(T) = \min \max f[T] \tag{1}$$

while $f[T]$ is estimated based on the previous traffic information, i.e

$$f[T] = g(f[T-1], f[T-2], \ldots, f[0]) \tag{2}$$

TABLE 2
Notations used in our work

Notation	Description
Parameters	
C_k	The capacity of a VM running VNF k, i.e. the amount of traffic that can be handled by an instance of VNF k
f_i	The traffic amount of SFC i
s_{ik}	Binary parameter to indicate if VNF k is included in SFC i
N_r	The number of available VM slots under rack r
S^i_{nk}	Binary parameter to indicate if VNF n is exactly before the VNF k in SFC i
Variables	
y_{rk}	Integer variable. The number of VNF instance k that is realized under rack r
p_{ir}	The fraction of SFC i that should be served under rack r
e_{ir}	The fraction of SFC i that enters rack r
d_{ir}	The fraction of SFC i that departures rack r
q^k_{ir}	The fraction of SFC i that receives the service of VNF k under rack r
x_{rk}	Binary variable to indicate if rack r hosts VMs for NF k. It is only used to formulate the cross-rack pipelined SFC model.

4.2 Formulation of VNF placement

In Section 3, we discussed four SFC deployment models that may be adopted in operator data centers. For simplicity, in this section, we only focus on the run-to-complete in a rack SFC deployment model and cross rack pipelined SFC model. However, the analyses can be easily applied to the server level placement model.

When we adopt the SFC deployment model as shown in Fig. 2(c), there is no cross-rack traffic, and hence, the computation resources is the only objective that should be optimized. Equivalently, we minimize the number of VMs set up for different types of VNFs. For the model shown in Fig. 2(d), there are always plenty of redundent VMs in the system. Hereby, we optimize the cross-rack traffic. Under this SFC deployment model, if all the racks are connected with nonblocking fabric, the cross-rack traffic should be always the same. Accordingly, we only discuss the case that all the racks are linearly connected. In this section, we will formulate the SFC placement problems under these two models, respectively.

4.2.1 SFC Run-to-Complete in a Rack

We first formulate the case that each SFC should be completed in a rack. In this case, there is no cross-rack traffic and hence we are to minimize the number of VMs set up for realizing VNF instances, i.e. the objective function is

$$\text{minimize} \qquad \sum_r \sum_k y_{rk} \tag{3}$$

where y_{rk} is the number of VMs set up for realizing VNF k. For the Run-to-Complete in a rack model, each SFC should be completed under the same rack. This can be formulated as

$$\sum_i p_{ir} f_i s_{ik} \le y_{rk} C_k, \quad \forall r, k \tag{4}$$

where p_{ir} is the fraction of SFC i that is to be completed in rack r, f_i is the traffic amount of SFC i, s_{ik} is a known binary constant to indicate if function k is included in SFC i, and C_k is the capacity of a VM if it realizes VNF k, i.e. the amount of traffic a VM can serve with VNF k.

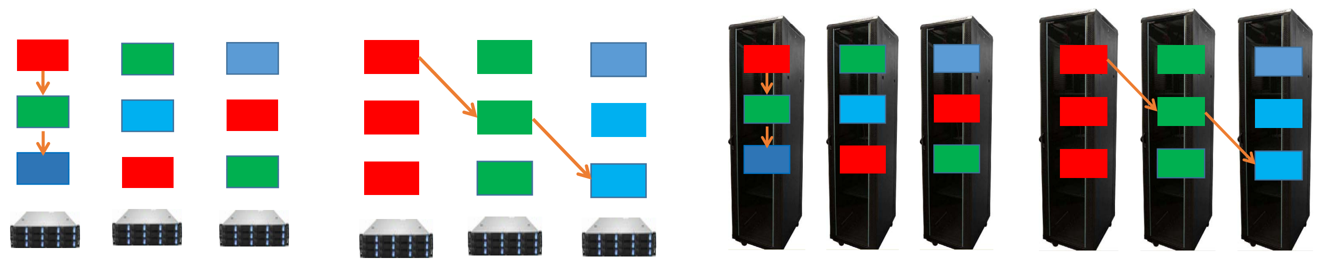

(a) SFC run-to-complete in a server. (b) Cross server pipelined SFC (c) SFC run-to-complete in a rack. (d) Cross rack pipelined SFC

Fig. 2. Typical service function chain deployment models.

As all the servers in a rack are connected by the ToR switch, and hence, we do not need to focus on the order of the network functions in each SFC. To ensure all the traffic of every SFC is served, there should be

$$\sum_r p_{ir} = 1 \quad \forall i \tag{5}$$

At last, there is another constraint to identify the number of available VM slots to realized VNFs under each rack:

$$\sum_k y_{rk} \le N_r \quad \forall r \tag{6}$$

where N_r is the number of available VM slots under rack r. Summing up above discussions, the VNF placement problem under SFC Run-to-Complete in a Rack Model (SRCR) can be formulated as

$$\boxed{\begin{array}{l}\text{minimize} \quad (3)\\ \text{Subject to:} \quad (4)\text{–}(6)\\ y_{rk} \text{ is integer}\end{array}}$$

4.2.2 Cross Rack Pipelined SFC

The other SFC deployment model is cross-rack pipelined SFC model, where each rack can only host one type of VNF and all the racks are linearly connected. Without loss of generality, we assume all the racks are linearly connected from rack 1 to rack R. In this case, we mainly focus on the traffic amount carried by the network. Assume e_{ir} and d_{ir} are the fractions of traffic for SFC i enters from rack r and departures from rack r, respectively, the amount of traffic for SFC i that is passing through the link between rack r and $r+1$ can be calculated as

$$f_i \sum_{t \le r} (e_{it} - d_{it})$$

Accordingly, the objective can be formulated as

$$\text{minimize} \quad \sum_i \sum_r [f_i \sum_{t \le r} (e_{it} - d_{it})] \tag{7}$$

Let q_{ir}^k denote the fraction of SFC i completes the function k at rack r,

$$\sum_{t \le r} q_{it}^n S_{nk}^i \ge q_{ir}^k, \quad \forall i, r, n, k \ne 0 \tag{8}$$

must be satisfied, where S_{nk}^i is a binary constant to indicate if VNF n is the previous network function of VNF k in SFC i. This constraint says that if VNF n and VNF k are two sequential network functions in SFC i, q_{ir}^k of SFC i can get service of VNF k at rack r, only if q_{ir}^k of SFC i has already gotten the service of VNF n before it enters rack r. For simplicity, we add a virtual function 0 at the beginning of each of the SFCs, and add a VNF M at the end of each of the SFCs, i.e. all the SFCs start from VNF 0 and end at VNF M. Based on this assumption, only the traffic enters network can receive the services of function 0, i.e.

$$\sum_{t \le r} e_{it} \ge q_{ir}^0, \quad \forall i, r \tag{9}$$

and all the traffic should complete its SFC requirement,

$$\sum_r q_{ir}^M = 1, \quad \forall i \tag{10}$$

In addition, the traffic can departure from each rack is limited by

$$\sum_{t \le r} q_{it}^M \ge d_{ir}, \quad \forall i, r \tag{11}$$

Above constraints (8) – (11) are used to formulate the SFC requirement. There are still some constraints for the VNF deployment. Such as the server capacity under each rack:

$$\sum_i f_i q_{ir}^k \le z_{rk} N_r C_k, \quad \forall r, k \tag{12}$$

where z_{rk} is a binary variable to indicate if rack r is to realize VNF k, and N_r is the number of available VM slots under rack r. Since every rack can host only one type of VNF, we have

$$\sum_{k \ne 0, M} z_{rk} \le 1, \quad \forall r \tag{13}$$

Based on above discussions, we can formulate the optimization problem for Cross Rack Pipelined SFC model (CRPS) as

$$\boxed{\begin{array}{l}\text{minimize} \quad (7)\\ \text{Subject to:} \quad (8)\text{–}(13)\\ y_{rk} \text{ is integer}\end{array}}$$

5 Traffic Estimation

In last section, we formulated the traffic forecasting and VNF placement problem that should be solved in our work. In the following two sections, we will discuss how to solve these formulated problems. In this section, we first discuss how to estimate the traffic amount upper bound in a short term, for example, 10 to 20 minutes later. With this forecasting, we can proactively configure the data center to adopt to the traffic amount, and hence not only keep service high availability, but also improve the network resource utilization.

The basic idea of traffic estimation in our work is slip-window linear regression. Given the traffic rate of previous k time instance, i.e. $f[i]$ for $i = t-k+1, t-k+2, \dots, t$, we can use

$$\hat{f}[t+1] = a_0 + \sum_{i=1}^{k} a_i f[t-i+1] \tag{14}$$

to estimate the traffic rate of time instance $t+1$. In this equation $f[t]$ is the real traffic rate at time instance t, while $\hat{f}[t]$ is the estimate traffic rate at time instance t. Based on our traffic measurement frequency, the time duration between two adjacent time instances is 20 minutes, which is enough for the data center reconfiguration.

With this estimation method, the most important question is how to determine the coefficient, i.e. a_i, in (14). There are two methods that can be considered. First, collect tons of data and training the model to derive a group of general coefficients, which is adopted in autoregressivemoving-average (ARMA) models. With this method, we can derive the traffic estimation soon. However, when the traffic characteristic changes, the coefficient may not work any more. More importantly, the traffic characteristic in the network changes from time to time. Accordingly, we adopt the second line of thought, determine the coefficient with the latest traffic measurement.

In this case, the training data can be formulated as following regression model:

$$Y = X\beta + \epsilon \tag{15}$$

where

$$Y = [f[t], f[t-1], \cdots, f[t-r+1]]$$

$$X = \begin{bmatrix} 1 & f[t-1] & f[t-2] & \cdots & f[t-k] \\ 1 & f[t-2] & f[t-3] & \cdots & f[t-k-1] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & f[t-r] & f[t-r-1] & \cdots & f[t-r-k+1] \end{bmatrix}$$

$$\beta = [a_0, a_1, \cdots, a_k]^T$$

$$\epsilon = [\epsilon_0, \epsilon_1, \cdots, \epsilon_k]^T$$

To minimize the estimation error, we should minimize $||\epsilon||$. It is well known that by setting

$$\beta = (X^T X)^{-1} X^T Y \tag{16}$$

we can get the minimal regression error [29]. At this time, the traffic estimation should be

$$\hat{Y} = X(X^T X)^{-1} X^T Y$$

and the estimation error is

$$\begin{aligned} Y - \hat{Y} &= Y - X(X^T X)^{-1} X^T Y \\ &= (I - X(X^T X)^{-1} X^T) Y \end{aligned} \tag{17}$$

From the (17), we know

- When X is square matrix, the estimation error is 0. However, this is due to the overfitting which should be avoided. Usually, there should be $r \gg k$ in practice.
- From the factor $X^T X$ in the estimation error expression, we know that when there is a larger traffic variance in the network, there will be larger estimation error. Therefore, we only apply such estimation on the aggregate traffic at each BRAS, rather than the traffic of individual subscriber.

It is worth noting that (14) with the parameters set as in (16) only estimate the average, rather than the upper bound, traffic rate in the next time instance, and hence it may underestimate the traffic amount injected into the network. In this case, we cannot provide required services to all the traffic, which impacts the service availability. To solve this problem, we should reserve some redundant capacity to ensure there are enough resources in case the real coming traffic amount is larger than the estimation.

To this end, we usually adopt the $3-\sigma$ principle to service high availability, i.e. reserve enough resources to provide services to the traffic whose amount is $f + 3\sigma$, where f is the average traffic rate, while σ is the traffic rate standard variance. However, as the traffic rate is distributed in a large range, this method may require too many resources. To save the amount of redundant capacity, we only consider the variance of the relative estimation error. Say $f(T)$ and $\hat{f}(T)$ are the real and estimated traffic rate, respectively, at time instance T. Then, we define the relative estimation error as:

$$e(T) = \max\{\frac{f(T) - \hat{f}(T)}{f(T)}, 0\} \tag{18}$$

Based on the fact that the latest traffic information provide more insight to the traffic variance in the short term future, we define the average estimation error as:

$$\bar{e}(T) = \frac{1-\alpha}{1-\alpha^N} \sum_{t=T-N+1}^{T} \alpha^{T-t} e(t) \tag{19}$$

where α is a parameter that between 0 and 1. Larger α means the history data is more important for traffic estimation. Accordingly, we usually assign to α with a number close to 0. Correspondingly, the traffic variance is defined as

$$v(T) = \frac{1-\alpha}{1-\alpha^N} \sum_{t=T-N+1}^{T} \alpha^{T-t} [e(t) - \bar{e}(t)]^2 \tag{20}$$

and the standard variance is

$$\sigma(T) = \sqrt{v(T)} \tag{21}$$

Based on above definitions, when we derive the average traffic estimation of time instance $T+1$, i.e. $\hat{f}(T+1)$, we estimate the traffic upper bound as

$$B(T+1) = \hat{f}(T+1)(1 + \bar{e}(T) + 3\sigma(T)) \tag{22}$$

In the other words, we reserve enough resources to provide service for so much traffic in case there is a unpredictable traffic burst.

6 Service Function Chain Deployment Algorithms

In Section 4, we formulated the problem we are to solve under different SFC deployment models. However, both models contain integer variables and hence they are difficult to solve in large scale data centers. In this section, we propose efficient algorithms to solve them.

6.1 Algorithm to Solve SRCR

At first, we design an algorithm to solve the SRCR problem. The key point to solve an Mixed Integer Linear Programming (MILP) is how to deal with the integer variables. A common method to deal with the integer variable is to relax the integer variables to be real variables, which derives a linear programming problem. Based on the solution of the linear programming problem, we can get some insights to yield the integer solution. By relaxing the integer variables, we can derive the following primal problem

Primal Model

$$\text{minimize} \qquad \sum_r \sum_k y_{rk}$$

Subject to:

$$\begin{aligned} & y_{rk}C_k - \sum_i p_{ir} f_i s_{ik} \geq 0, \quad \forall r, k \\ & \sum_r p_{ir} \geq 1 \quad \forall i \\ & -\sum_k y_{rk} \geq -N_r \quad \forall r \\ & y_{rk}, p_{ir} \geq 0 \quad \forall i, r, k \end{aligned}$$

One possible method to derive an integer solution is that we first solve the primal model, which may derive a fractional solution of y_{rk}, and then round up the fractional solution to be the nearest integer. However, this method may suffer from infeasible solution which may not satisfy the constraint $\sum_k y_{rk} \leq N_r$.

Another thought is to consider which rack is the most valuable to host a new VM. If rack r is the most valuable to host VNF k, more flows requiring VNF k should be steered to rack r, which results in $y_{rk}C_k - \sum_i p_{ir} f_i s_{ik} = 0$. This indicates the dual variable of constraint $y_{rk}C_k - \sum_i p_{ir} f_i s_{ik} \geq 0$ should be positive. The larger dual variable means the corresponding assignment, i.e. assign VNF k under rack r, can reduce the objective function. Following this line of thought, we consider the dual problem:

Dual Model

$$\text{maximize} \qquad \sum_i x_i - \sum_r N_r u_r$$

Subject to:

$$\begin{aligned} & C_k z_{rk} - u_r \leq 1, \quad \forall r, k \\ & x_i - f_i \sum_k s_{ik} z_{rk} \leq 0 \quad \forall i \\ & z_{rk}, x_i, u_r \geq 0 \quad \forall i, r, k \end{aligned}$$

where z_{rk}, x_i and u_r are the dual variables associated with the constraints in the primal model, respectively. Consider the fact that we usually do not fully utilize the capacity of a specific rack, and hence the constraint $\sum_k y_{rk} \leq N_r$ should always be loose. Accordingly, we have $u_r = 0$ in the dual model. To maximize the objective of dual model, we should increase the value of x_i, while ensuring that there is z_{rk} to make the problem feasible. Therefore, the main idea of our algorithm is to increase x_i in the dual model, till there is only a single z_{rk} that can keep the model feasible, i.e. we cannot continue increasing the x_i, otherwise, there will be no feasible solution. In this case, the rack r associated with the largest $\sum_k z_{rk}$ (Since a flow should traverse multiple VNFs, we should find the rack with the largest summation over all the related VNFs.) is exactly the rack that is the best one to accommodate the traffic of SFC i. By fixing the rack to accommodate the traffic of SFC i, we assign as much traffic for SFC i as possible into the selected rack. After each above iteration, we update the network capacity and continue the next iteration to find another rack to accommodate SFCs and assign traffic to the selected rack.

Algorithm 1: Primal-Dual based algorithm for SRCR.

```
Input: Traffic amount $f_i$ and its SFC requirement $s_{ik}$
Output: VNF placement $y_{rk}$ and traffic assignment $p_{ir}$
1: Formulate the primal and dual problems based on the input, and initialize $c_i \leftarrow 0$, $R_i \leftarrow \Phi$, $z_{rk} \leftarrow 0$, $p_{ir} \leftarrow 0$ and $y_{rk} \leftarrow 0$
2: while not all $c_i = 1$ do
3:     Find out an $i$ such that $c_i < 1$
4:     Continuously increase $x_i$ till there is only a single feasible solution in dual model
5:     $r' \leftarrow \arg\max_{r \notin R_i} \sum_k z'_{rk}$, where $z'_{rk}$ is the solution of dual model with the fixed $x_i$
6:     Find out the maximum amount of traffic for SFC $i$, that can be accommodated by the residual capacity of rack $r'$, say it is $p'_{ir'}$.
7:     $p_{ir'} \leftarrow \min\{1 - c_i, p'_{ir'}\}$, $c_i \leftarrow c_i + p_{ir'}$, $R_i \leftarrow R_i \cup r'$
8:     Update $y_{r'k}$ based on the traffic assignment
9: end while
10: return $y_{rk}$ and $p_{ir}$
```

Based on above discussions, we propose Algorithm 1 to solve SRCR model. In this algorithm, S_i is the set of all the network functions in SFC i.

In Algorithm 1, c_i is used to record the fraction of traffic of SFC i that has already been assigned to a rack, while R_i is the set of racks that have been used to accommodate traffic of SFC i. When c_i is less than 1, we should continue to find out a rack to provide services to the traffic with requirement of SFC i. To this end, we increase x_i and ensure there is z_{rk} to keep the feasibility of the dual model. The largest x_i can be found with binary search, and the feasibility can be checked with simplex algorithm (Line 4). When the largest x_i with feasible solution is found, we assign as much traffic of SFC i as possible to the rack r, such that $r' \leftarrow \arg\max_{r \notin R_i} \sum_k z'_{rk}$, where z'_{rk} is the solution of the dual model with the fixed x_i (Line 6), since the largest dual variable indicates the most valuable rack to accommodate flow i. After that, we update corresponding parameters in Line 7. This algorithm ends when all the traffic is assigned to some racks and fulfill the SFC requirement. In this algorithm, every iteration should deal with some of the traffic, still all the traffic recieves the service of its required SFC. Accordingly, this algorithm will converge in the end.

It is worth noting that though we adopt the primal-dual method, which is a well-known technology, to solve the SRCR. However, we do not directly leverage any existing algorithm since the primal-dual technology is developed for the LP problem, rather than ILP problem as in our work. We only leverage the algebra meaning of primal-dual technology to derive some insights, such that we can efficiently assign the SFCs to different racks.

6.2 Algorithm to solve CRPS

In the CRPS, similar to the SRCR problem, the main ingredient makes the problem intractable is the integer variable z_{rk}. Again, we can relax z_{rk} to be real variable, and then CRPS will become linear programming (LP) problem which is easy to solve. After relaxation, however, we cannot adopt the same primal-dual technology as to solve CRPS, since there are too many constraints in the CRPS problem and it is difficult for us to design a primal-dual based algorithm. Consider z_{rk} is a special type of integer variable, i.e. binary variable, we can take another method, relaxation and rounding to solve CRPS.

The basic idea of relaxation and rounding algorithm is that we first relax the binary variable in CRPS and get an LP model. By solving this relaxed LP model, we can get a solution in which z_{rk} can be a fractional value between 0 and 1. Accordingly, we have to derive a binary solution based on such fractional solution. For example, if z_{11} is the largest variable among $\{z_{1\cdot}\}$, we can directly set $z_{11} = 1$ and $z_{1k} = 0$ for all $k \neq 1$.

However, this rounding method may not derive a good solution, since when we round a variable z_{lk}, we neither consider the impact of its rounding to other variables, nor how the rounding of other variables impacts z_{lk}. To yield a better solution, we do not round all the z_{rk} at one time. When some of the binary variables are fixed, the polyhedron of solution space will corresponding changes, and hence, change the extreme point, i.e. the optimal choice of the rounding of the other variables. Based on this line of thought, we adopt progressive rounding method in our algorithm. In each iteration, we only round part of the variables in each iteration. Then, we solve the LP model once more, and try the next rounding procedure, till all the variables are rounded to be binary value.

Based on above discussions, we design a progressive relaxation and rounding algorithm to solve CRPS, which is shown in Algorithm 2. In each iteration (Lines 2–10), we first solve the relaxed CRPS (Line 3). If the relaxed CRPS is infeasible, the algorithm returns an empty solution (Lines 4–6). This is becasue that there is no feasible solution for the relaxed problem, let along the orginal ILP model. If a feasible solution is obtained, we find a rack to fix the VNF type it hosts (Lines 7 and 8) based on the above discussions. After that we fix the variable associating with the fixed VNF type, and resolve the relaxed CRPS. The iteration procedure ends when the type of the VNF instances hosted by every rack is fixed. At last, we solve the relaxed CRPS once more to calculated how to steer the traffic to realize all the SFC.

Algorithm 2: Relaxation and Rounding for CRPS

```
Input: Traffic amount $f_i$ and its SFC requirement $s_{ik}$
Output: VNF placement $z_{rk}$ and service providing scheme $q_{it}^k$
1: Formulate CRPS model according to the input
2: while not all $z_{rk}$ have binary value do
3:     Solve relaxed CRPS
4:     if the problem is infeasible then
5:         return $\Phi$
6:     end if
7:     Find the rack $r$ and function $k$ such that
       $(r, k) = \arg\max_{r,k} z_{rk}$
8:     $z_{rk} \leftarrow 1$ and $z_{rk'} \leftarrow 0$ for all $k' \neq k$
9:     add more constraints into the relaxed CRPS
10: end while
11: Solve CRPS model with all fixed $z_{rk}$ to derive the service
    function providing scheme
12: return $\{z_{rk}\}$ and $q_{it}^k$
```

7 Practical Issues and Implementation

7.1 Practical Issues

To realize the dynamic VNF scaling in operator data centers, there are some practical issues. The first one is that we do not consider the cost of VM migration. It is really possible that two sequential time instances should be implemented as absolutely different configurations even if there is only a very little traffic variance. To prevent large scale VM migration, we can add some constraints in the SRCR model and CRPS model to fix some of the VNF placement. For example, when the traffic rate increases, we can fix all the configurations in previous time instance and optimize the incremental traffic. If the traffic rate decreases, we can only simply scale out some of the VMs. Periodically, we can leverage more VM migrations to optimize the data center resource utilization. For example, we only fix the configuration of the highly utilized VMs/racks, and then optimize the configurations of the remaining data center.

Another practical issue is that our traffic rate estimation algorithm should be applied to the aggregate traffic. There are two ways to collect the history data that is required for the forecasting purpose. First, we can collect the traffic rate of each SFC at the BRAS. When there are thousands of SFCs in the network, it may be a large overhead for the BRAS. Alternatively, we first calculate the percentage of the traffic that requires each type of SFC in the network, and only forecast the traffic rate of total aggregate rate, rather than the traffic of each SFC. In this case, we can greatly reduce the overhead to collect traffic rate. In addition, if all the SFC services are provided at remote data centers which are far from the BRAS, we can also collect the traffic measurements at the aggregate routers or even the core routers to reduce the measurement cost and get a more stable aggregate traffic rate, which is benefit to the traffic estimation.

7.2 Implementation

In our operator network, we implement a VNF pooling system as shown in Fig. 4. All the VNFs are managed as a VNF pooling system in data centers. In this VNF pooling system, all the VNFs are managed based on the ETSI VNF framework. Traffic from the subscribers should first be aggregated by the BRASes and injected into the VNF pooling system. Based on the type of subscribers, access traffic should pass through different SFCs, and then be injected into the Internet.

To manage the VNF resource pool in our data center, we deploy three types of controllers. At first, Virtualized Infrastructure Manager (VIM) is used to manage the Network Function

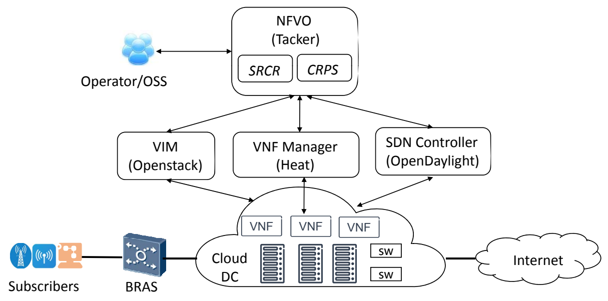


Fig. 3. Implementation in real systems.

Virtualization Infrastructure (NFVI). For example, VIM should work on starting up, initializing and updating VMs. In our system, the VIM is realized by Openstack.

VNF Manager (VNM) works on not only managing live period of each VNF, but also dynamically scaling in and scaling out VNF instances. In our system, VNFM is implemented by Openstack Heat 6.0.0.

SDN controller configures the ToRs or virtual switches. It steers the traffic in the network to pass through required SFC by controlling the flow table in the ToR switches or virtual switches. In addition, the SDN controller collects the traffic information in the system. In our current system, the SDN controller is based on OpenDaylight.

To orchestrate the function of all above three types of controllers, an NFV Orchestrator (NFVO) is implemented in the system. In the NFVO, we install the SRCR and CRPS algorithms. When the SDN controller gets the traffic measurement, it reports to the NFVO. NFVO triggers the SRCR or CRPS algorithm to calculate how many VNF instances of each type should be set up under each rack and sends the results to VIM and VNFM. In addition, the NFVO also derives the traffic routing scheme and advertise it to the SDN controller. In our system, the NFVO is implemented with tacker technology.

8 Experimental Results

In this section, we study the performance of our system through both testbed experiments and large scale real trace driven simulations. At first, we implement a real VNF named IP Multimedia Subsystem (IMS) and test the performance of our dynamic VNF scaling system. Since the traffic estimation algorithm greatly impacts the performance of our system, we investigate the estimation accuracy in Section 8.2. Then, we discuss the performance of algorithms to dynamically scale the VNFs under SFC Run-to-Complete in a Rack model and Cross Rack Pipelined SFC model in Section 8.3 and 8.4, respectively.

To test the universality of our algorithms, we collect traffic data from 4 BRASes located at different cities in Guangdong province of China. The traffic collection last for 2 days and we measure the traffic rate of each BRAS every 20 minutes. As we discussed in Section 3.1, it is difficult to forecast the traffic rate of a specific subscriber, we apply our algorithms in the BRAS level.

8.1 Testbed Experiment

Fig. 4 shows the realization of our operator system. Since China Telecom is a Chinese operator, the User Interface (UI) is in Chinese. However, we translate the key words in the UI. Fig. 4(a) shows how we scale in a VNF. To scale in a VNF instance, we should set the parameters of the VNF to indicate where to scale in such VNF instance and which type of this VNF instance is. Fig. 4(b) shows that how our system indicates a VNF instance should be scaled in due to the VNF high utilization (the red ones).

To study the performance of our VNF scaling algorithms in real system, we implement a VNF named IMS. Since IMS is actually including a service function chain, we adopt the run-to-complete in a rack model in the testbed experiment. Based on the traffic amount forecasting, we dynamically scale in more IMS instances into the system when we expect there is more traffic that will be injected into the system, and scale out some of the instances when we expect a decreasing amount of traffic in the next time instance. Compared with deploying the fixed number of IMS instances to accommodate the peak traffic amount, the average number of IMS instances required to serve traffic is reduced by about 18%. In the meanwhile, the service failure rate keeps almost the same.

8.2 Performance of Traffic Estimation

In this section, we investigate the performance of our traffic estimation algorithms. We will investigate the performance in three perspectives: 1) the accuracy of the estimation; 2) the performance to handle all the traffic, i.e. times of traffic overflow during the test period; 3) the amount of resources required to serve all the traffic, which is presented as the amount of traffic that the reserved resource should be enough to serve. As a baseline for the performance of 2) and 3), we also leverage the 3-σ principle to calculate the amount of traffic that we should reserve resources for in order to keep the high service availability.

8.2.1 Estimation Accuracy

In this section, we define the average estimation error as

$$\bar{E} = \sum_t E(t) \tag{23}$$

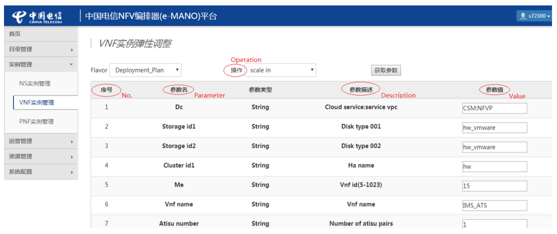
with

$$E(t) = \frac{|f(T) - \hat{f}(T)|}{f(T)} \tag{24}$$

where $f(T)$ and $\hat{f}(T)$ are the real and estimated traffic rate, respectively. As the traffic estimation performance is mainly impacted by two parameters, the regression order (i.e. k in (14)) and the size of training set (i.e. r in (15)), we study how the estimation performance changes with these two parameters.

Fig. 5 shows how the estimation error changes with these two parameters. By fixing the training set size to be 15 and changing the regression order, we derive the results shown in Fig. 5(a). From this figure, we can see that the estimation error is reducing with the increase of regression order. This is an intuitive phenomenon since higher regression order provides larger freedom degree to the regression model to trace the traffic trend.

In Fig. 5(b), we fix the regression order to be 5 and change the training set size. Intuitively, the larger training set size would lead to a more accurate estimation. However, it is not the case in our simulation. From this figure, we observe that the estimation error is first reducing with the training set size, and then increasing

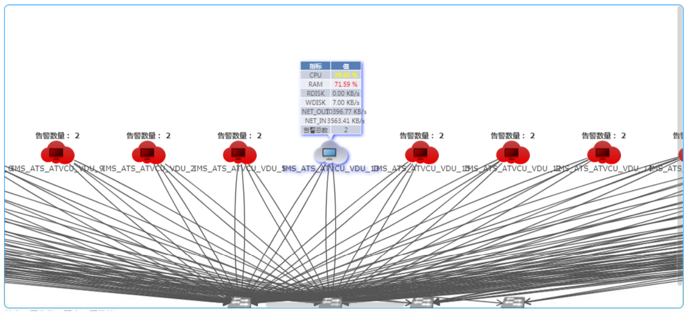


VNF实例弹性调整

Flavor: Deployment_Plan   Operation 操作: scale in

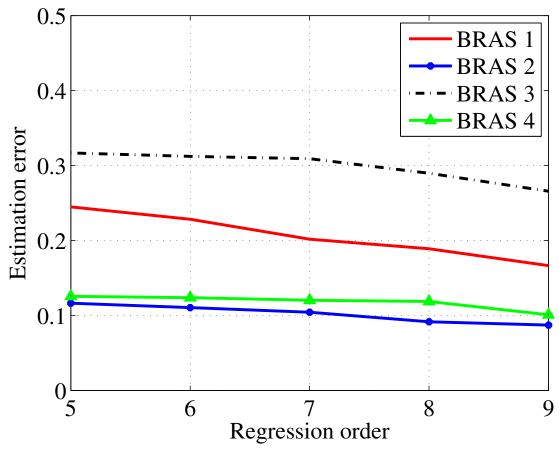
序号 No.	参数名 Parameter	参数类型	参数描述 Description	参数值 Value
1	Dc	String	Cloud service:service vpc	CSM:NFVP
2	Storage id1	String	Disk type 001	hw_vmware
3	Storage id2	String	Disk type 002	hw_vmware
4	Cluster id1	String	Ha name	hw
5	Me	String	Vnf id(5-1023)	15
6	Vnf name	String	Vnf name	IMS_ATS
7	Atisu number	String	Number of atisu pairs	1

(a) Realization of MANO system.

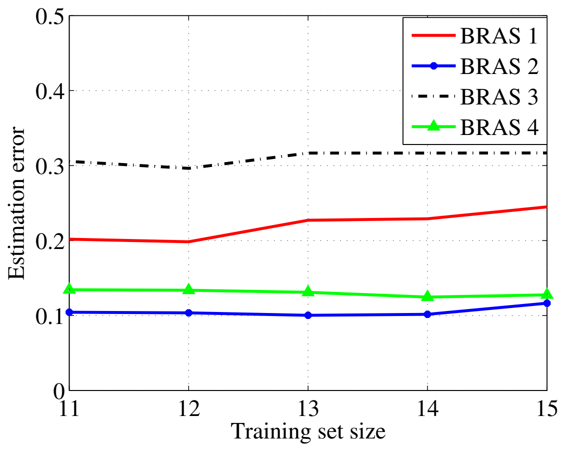


(b) Utilization alarm system

Fig. 4. Real system implementation.



(a) Estimation error vs. regression order.



(b) Estimation error vs. training set size

Fig. 5. Estimation error.

with the training set size. This is because that the traffic rate characteristic is varying during the time. The traffic changing information in the long history may provide negative information to train our forecasting model. Therefore, large training set may not always improve our forecasting accuracy. This demonstrates that we cannot find a uniform parameter for the forecasting model, and we should dynamically change the parameters for an accurate traffic forecasting. Through plenty of simulations, we find that for a given regression order, set the training set size to be 12–14 can derive the best traffic forecasting performance.

8.2.2 *Performance to Handle All Traffic*

Due to the high service availability purpose, we should reserve redundant resources in case there is a traffic burst. In our algorithm, we use (22) to estimate the traffic rate upper bound which is the amount of traffic that we should reserve enough resources for. To see if (22) can help us to reserve enough capacity, we set the regression order as 9, the training set size to be 12, and test the number of time instances in which our algorithm and 3-σ principle would underestimate the traffic rate during the 60 test time instances. In addition, we also test the case that we do not reserve any redundant resources based on the linear regression model (labeled as "Only LR"). This can show the necessity of deploying redundant resources in our forecasting algorithm. The simulation results are shown in Tab. 3.

TABLE 3
Time instances of traffic underestimation

	With (22)	With 3-σ principle	Only LR
BRAS 1	0	2	37
BRAS 2	0	3	39
BRAS 3	1	5	41
BRAS 4	1	2	26

From this table, we can see that at most of the time instances, the "only LR" scheme underestimate the traffic amount. This is due to two main reasons. Firstly, linear regression is to estimate the average traffic rate, which does not consider the estimation is larger or less than the real value; second, there may be traffic bursts which cannot be handled by the "only LR" scheme.

When we add some redundant traffic amount based on (22), we can greatly reduce the number of time instances at which the traffic amount is underestimated. This demonstrates that our forecasting method can deal with the forecasting error and hence ensure the high service availability.

Compare the performance of the two schemes with redundant

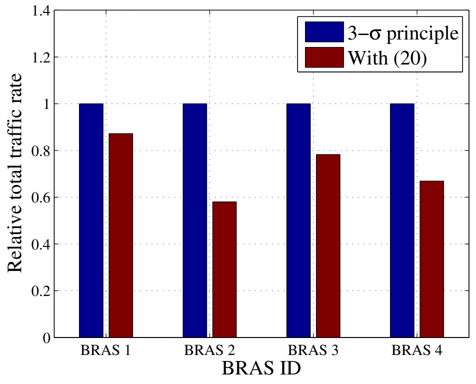


Fig. 6. Traffic amount that should be served by the reserved resources.

resources, there are fewer number of time instances in which traffic amount is underestimated with (22) than that with 3-σ principle. We further dig into the simulation, we find that whenever the traffic underestimation occurs with (22), it also occurs with 3-σ principle. This is because that (22) applies the 3-σ principle on the error part of the estimation, rather than the entire traffic, it can better track the traffic rate variance.

8.2.3 Amount of Resources Reserved in the System

Though (22) can help us provide high availability, it may cost more resources. Fig. 6 shows the relative total traffic rate we should provision resources to serve during the 60 test time instances. In this figure, we normalize the total traffic rate with the larger one of each BRAS. In the other words, for each BRAS, the scheme that should reserve resources for larger total traffic rate should reserve resources for 1 unit of relative total traffic rate.

From this figure, we can see that (22) requires about 10%-40% less resources in order to handle the traffic burst and provide high service availability. This is again due to it applies the 3-σ principle only on the error part of the traffic estimation and tracks the traffic rate variance better. Compare Fig. 6 with 5, we can see that more accurate traffic forecasting results in larger resource saving. It indicates that if we leverage larger regression order, we can save more resources in order to ensure high availability.

8.3 VM Number under Run-to-Complete in a Rack Model

In the following two subsections, we are to investigate the performance of our dynamic VNF scaling algorithms. As there are still no available service chain requirement data in real operator networks, we assume there are 10 ordered service functions and each SFC randomly requires 4 to 6 of these service functions. Therefore, there are 672 different SFCs in the network. We further assume all the traffic evenly require these SFCs, i.e. every 1/672 of the traffic from the BRAS requires the same SFC. In the simulation, we normalize the traffic rate on to [0,672], and hence the traffic requires each SFC is distributed on [0,1]. In addition, to provide a network function to a unit of traffic, it requires [20,40] VNF instances according to the network function type.

For the comparison purpose, we propose two baselines: 1) to study the performance of our algorithm, we use the real traffic rate as input, and compare the number of VNF instances required by our algorithm with that required by the naive first-fit algorithm, i.e. put as much of every flow as possible into the rack that can serve part of it, it there is not enough VNF instances, scale in new VMs to provide the required SFC; 2) to study how many VNF instances are redundant in order to guarantee the system high availability, we employ our algorithm to calculate the required number of VNF instances with the forecasted traffic and the real traffic as input, respectively.

The simulation results are shown in Fig. 7. In this figure, we can make three main observations. First, by comparing the performance of Algorithm 1 with the greedy VNF instance scaling method, we can see that see that the Algorithm 1 needs about 30% less VNF instances than that required by greedy based algorithm. This is because Algorithm 1 optimizes the traffic splitting and VNF placement in the global view, while the greedy algorithm only pursues the minimal VNF instance number by considering current SFC requirement but not the future requirement.

Second, it requires about 70% more VNF instances by using the forecasted traffic amount upper bound as the algorithm input than that derived by using the real traffic as the algorithm input. By carefully comparing curve derived by using the forecasting traffic and that derived by using the real traffic, we can find that the peaks on the curve derived by using the forecasting traffic has a group delay of 1 or 2 time instances. Accordingly, more VNF instances should be reserved to handle the traffic increasing and traffic burst scenarios.

Last but not the least, when the traffic rate is changing relatively smoothly, the VNF instances number required by the algorithm with forecasting traffic upper bound as input is close to that required by using the real traffic as input. This observation is not only due to the accuracy of our traffic forecasting algorithm, but also because the VNF instance quantize the traffic amount and reduce the VNF instance number gap between above two cases.

8.4 Total Bandwidth consumption under Cross Rack Pipelined Model

With the same demand settings as in last subsection, we investigate the performance of the Algorithm 2 to save the network bandwidth. It should be noted that all the bandwidth consumption in the simulation results are calculated based on the real traffic requirement with different VNF placment and traffic steering schemes derived by different algorithms, rather than the forecasted traffic requirement nor the amount of traffic that can be supported by the network. The simulation results are shown in Fig. 8. From this figure, we can see that by implementing Algorithm 2 with the real traffic amount as input, we can derive the best performance in all cases, as Algorithm 2 optimizes the bandwidth consumption in a global view. However, there is a very interesting observation that the performance of Algorithm 2 with the forecasted traffic rate as input is better than the performance of greedy algorithm with real traffic rate as the input. This is because that the forecasted traffic has the same trend as the real traffic. If we proportionally increase the traffic of each SFC requirement, since the VM capacity under a rack is very large compared with the requirement of a single SFC requirement, Algorithm 2 will derive almost the same VM placement and traffic routing scheme as we leverage the real traffic amount as input. This simulation

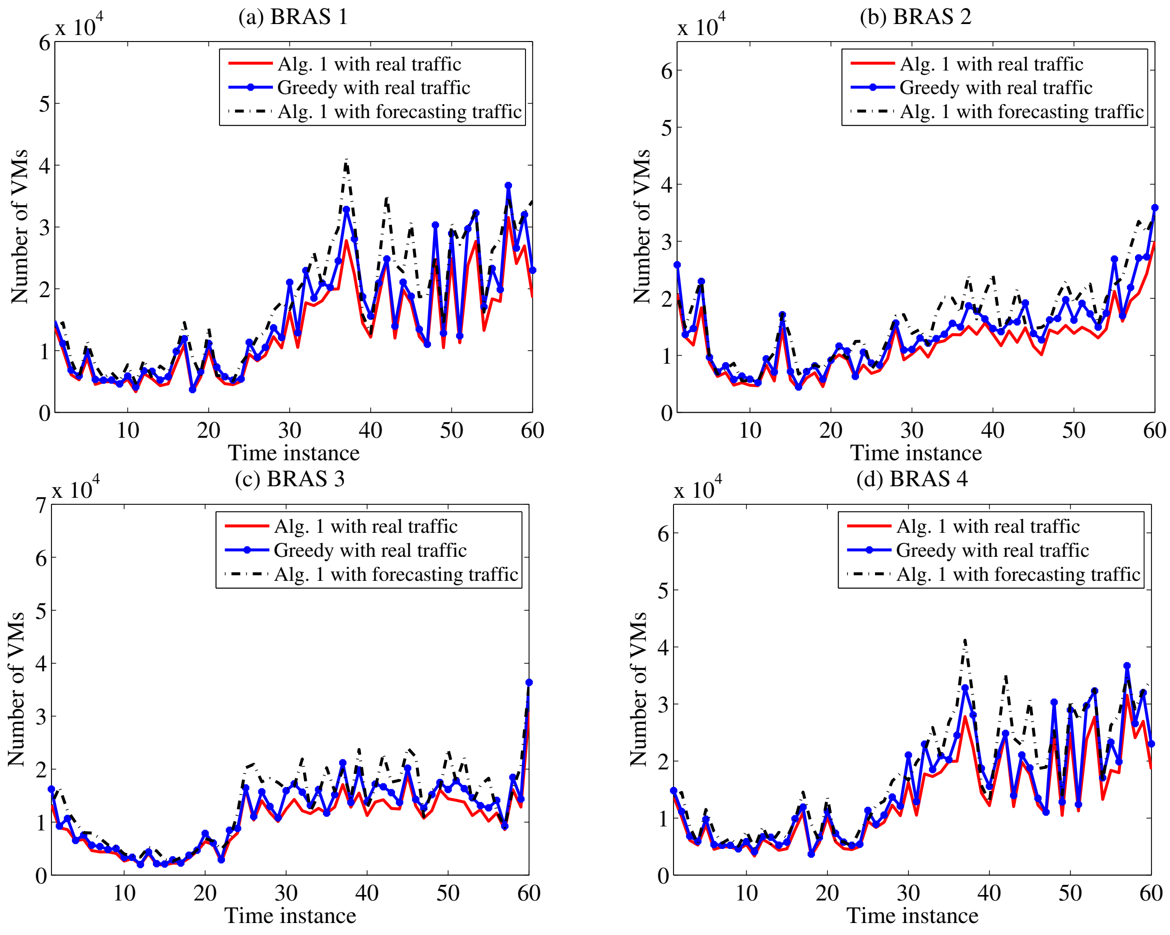


Fig. 7. VM numbers in SRCR.

shows the robustness of Algorithm 2 when we cannot accurately forecast the traffic rate.

It is worth noting that, in both Fig. 7 and 8, it seems that the peaks on the curve derived with forecasted traffic amount appear at the same time instances as the curve derived with the real traffic. However, this is not the case. In fact, the peak on the curve derived with forecasted traffic amount appear one or two time instances later compared with the curve derived with the real traffic, which is clearer in Fig. 7(c). This is due to the fact that we cannot know there will be coming traffic bursts in advance. When we observe a traffic burst, we expect there will be more traffic in the network, and hence, the forecasted traffic amount increases.

In addition, we may note that the trend of the curves in Fig. 7 are very similar to those in Fig. 8. This is because that the simulations are based on the traffic from the same BRASes. More traffic results in more VNF instances in run-to-complete in a rack model while more cross-rack traffic in pipelined SFC model. Accordingly, the curves in Fig. 7 and Fig. 8 apply to the same trend.

8.5 Algorithm Running Time

As an online system, the algorithm running time is an important issue. In this section, we study how the running time of all the three algorithms proposed in our work changes with the simulation scale. All the simulation results are collected from a Dell desktop carrying Intel i7-2600 CPU with 8 GB memory. Since the algorithm running time may vary with the traces, Every point in the simulation results is averaged by 100 tries. The simulation results are shown in Fig. 9.

Fig. 9(a) shows how the time to forecast traffic amount changes with the number of training set size and regression order. Apparently, the time to forecast traffic amount should increase with the number of training set size and regression order, since we should calculate the inverse and multiplication of larger matrices. In addition, the running time increases faster and faster with these two parameters. However, to derive an acceptable forecast solution, i.e. with the regression order 9 and the training set size 12, the time for forecasting traffic amount of each time instance is only tens of seconds, which is much less than the time for scaling in a VNF instance.

Fig. 9(b) shows how the running time of Algorithm 1 changes with the number of SFCs and racks in the system. From this figure, we can make two observations. Firstly, the algorithm running time is linearly increasing with the number of SFCs in the system, since Algorithm 1 can deal with only one SFC in each iteration. Secondly, the number of racks in the system determines the scale of LP model that we should check its feasibility. Accordingly, the algorithm running time is increasing faster and faster with the number of racks. To determine the VNF placement and traffic routing under the SRCR model, we need several seconds. Plus the time to forecast the traffic amount, the total algorithm running time is much less than the duration of a time instance, and hence

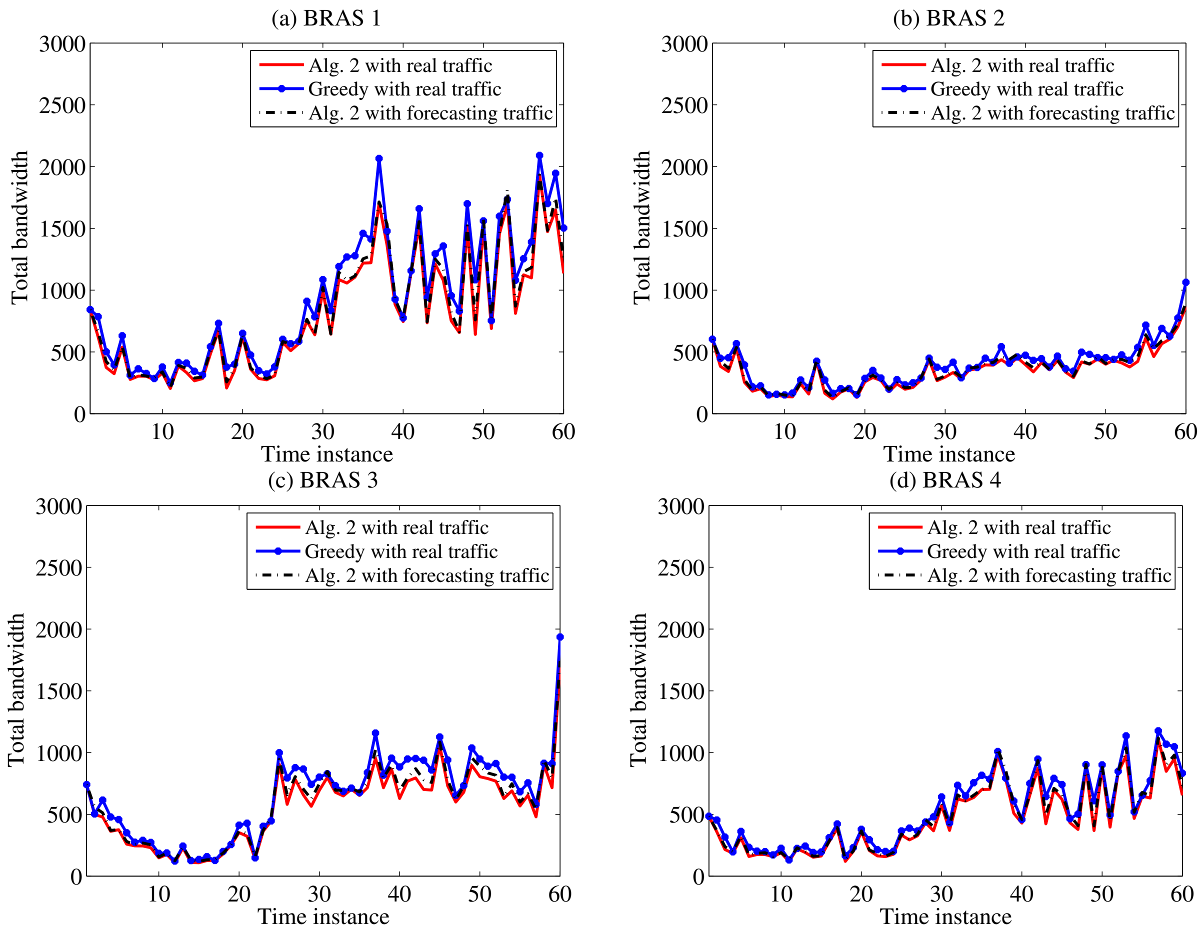


Fig. 8. Relative bandwidth requirement in CRPS.

our algorithm is quick enough for an online system under SRCR model.

Compared Fig. 9(c) with Fig. 9(b), we find that Algorithm 2 needs several minutes to determine the VNF placement. This is because in Algorithm 1, we only need to check the LP feasibility, while we need to find the optimal solution in Algorithm 2. However, several minutes is also reasonable in our system since the length of a time instance usually larger than 10 minutes.

9 Conclusion

In this paper, we introduced a dynamic VNF instance scaling system in our real operator network. We analyzed the traffic characteristics in our operator network and introduced how the VNFs are organized to provide service chains in real operator system. Based on these facts, we first propose an algorithm to forecast the traffic rate upper bound in the near future. Based on this traffic forecast, we propose algorithm to proactively scale in VNF instances in order to keep the service high availability. Both implementation in real operator network and extensive real data driven simulations show that dynamic VNF scaling can greatly save the resources required to serve all the traffic.

Acknowledgment

The authors would like to thank anonymous reviewers for their valuable comments.

References

[1] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, “Simple-fying middlebox policy enforcement using sdn,” in *ACM SIGCOMM CCR*, 2013.

[2] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, “Toward software-defined middlebox networking,” in *ACM Hotnets 2012*.

[3] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, “Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags,” in *Proc. USENIX NSDI*, 2014.

[4] “Network functions virtualisation – an introduction, benefits, enablers, challenges & call for action,” in *ETSI NFV ISG*, 2012.

[5] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, “Split/merge: System support for elastic execution in virtual middle-boxes.” in *NSDI*, 2013, pp. 227–240.

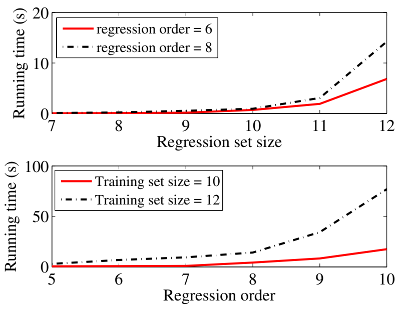
[6] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, “Opennf: Enabling innovation in network function control,” in *Proceedings of the SIGCOMM 2014*.

[7] S. Palkar, C. Lan, S. Han, K. Jang, A. Panda, S. Ratnasamy, L. Rizzo, and S. Shenker, “E2: A framework for nfv applications,” in *ACM SOSP*, 2015.

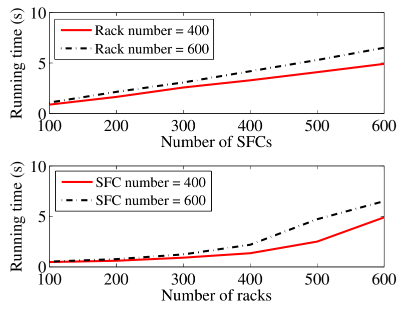
[8] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, A. Akella, and V. Sekar, “Stratos: A network-aware orchestration layer for middleboxes in the cloud,” *CoRR*, vol. abs/1305.0209, 2013. [Online]. Available: http://arxiv.org/abs/1305.0209

[9] A. Dwaraki and T. Wolf, “Adaptive service-chain routing for virtual network functions in software-defined networks,” in *Proceedings of the ACM HotMiddleBox*, 2016, pp. 32–37.

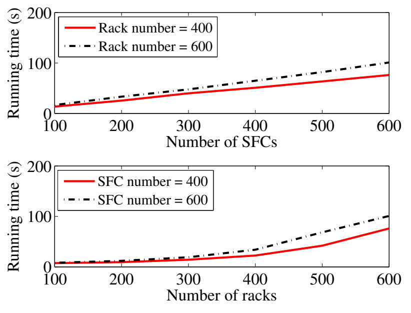
[10] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, “Softnic: A software nic to augment hardware,” EECS Department, University of California, Berkeley, Tech. Rep., May 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-155.html



(a) Forecasting algorithm.



(b) Primal-dual algorithm for SRCR.



(c) Relaxation and rounding for CRPS.

Fig. 9. Algorithm running time.

[11] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, “Netbricks: Taking the v out of nfv,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016, pp. 203–216.
[12] J. Hwang, K. K. Ramakrishnan, and T. Wood, “Netvm: High performance and flexible networking using virtualization on commodity platforms,” *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, March 2015.
[13] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, “Clickos and the art of network function virtualization,” in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, 2014, pp. 459–473.
[14] W. Zhang, G. Liu, W. Zhang, N. Shah, P. Lopreiato, G. Todeschi, K. Ramakrishnan, and T. Wood, “Opennetvm: A platform for high performance network service chains,” in *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, pp. 26–31.
[15] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, “Design and implementation of a consolidated middlebox architecture,” in *USENEX NSDI 2012*.
[16] A. Abujoda and P. Papadimitriou, “Midas: Middlebox discovery and selection for on-path flow processing,” in *COMSNETS*, 2015.
[17] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions in nfv,” *arXiv preprint arXiv:1503.06377*, 2015.
[18] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghayi, D. Li, G. Wilfong, Y. R. Yang, and C. Guo, “Pace: Policy-aware application cloud embedding,” in *IEEE INFOCOM*, April 2013.
[19] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, “A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining,” *Comput. Commun.*, vol. 102, no. C, pp. 67–77, Apr. 2017.
[20] S. Draxler, H. Karl, and Z. A. Mann, “Joint optimization of scaling and placement of virtual network services,” in *IEEE CCGrid*, 2017, pp. 1–10.
[21] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” in *IEEE INFOCOM 2010*.
[22] M. Jiber, I. Lamouik, Y. Ali, and M. A. Sabri, “Traffic flow prediction using neural network,” in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, April 2018, pp. 1–4.
[23] T. H. H. Aldhyani and M. R. Joshi, “Integration of time series models with soft clustering to enhance network traffic forecasting,” in *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Sept, pp. 212–214.
[24] L. Tang, S. Du, and S. Ji, “Forecasting network traffic at large time scale by using dual-related method,” in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Aug, pp. 1336–1340.
[25] X. Zhang, C. Wu, Z. Li, and F. C. M. Lau, “Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization,” in *IEEE INFOCOM 2017*, pp. 1–9.
[26] X. Fei, F. Liu, H. Xu, and H. Jin, “Adaptive vnf scaling and flow routing with proactive demand prediction,” in *IEEE INFOCOM 2018*, pp. 1–9.
[27] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.
[28] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “Vl2: A scalable and flexible data center network,” in *Proceedings of the ACM SIGCOMM 2009*.
[29] J. Rawlings, *Applied regression analysis: a research tool*, ser. Wadsworth & Brooks/Cole statistics/probability series. Wadsworth & Brooks/Cole Advanced Books & Software, 1988. [Online]. Available: https://books.google.com/books?id=PRnvAAAAMAAJ

Hong Tang is Ph.D. candidate of School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China. He is also a network architect in Guangzhou Institute of China Telecom. And his main research interest is traffic planning inIP backbone network and traffics scheduling in Software Defined Network and Network Function Virtualization. He received his BSc from Huazhong University of Science and Technology, China, in 1997. And he received his MSc in optical engineering from Jinan University, China, in 2003.

Danny Zhou was Sr. SDN/NFV software architect from Intel Network Platforms Group. He led an industry task force successfully upstreamed NSH (Network Service Header) to OpenDaylight, OpenvSwitch and FD.io open source communities, which not only paves the road for NSH to IETF standard, but also contributes key ingredients of open source NSH-based SFC solution in NFV environment for appropriate Telco use cases such as GiLAN. Danny published 9 international patents and patent applications around NFV, and delivered 9 presentations on various industry events summits. Danny Zhou joined Huawei in 2018, and now he is working at Huawei’s Products and Solutions Group as specialist focusing on chip planning and architecture design for Cloud Networking and Network Function Virtualization.

Duan Chen is a senior strategy manager on network architecture and IP transport network in Nokia Shanghai Bell Co., Ltd. He received his B.S. degree from Huazhong University of Science and Technology in Wuhan, China, and M.S. degree from Xian Jiaotong University in Xian, China. Currently he is the vice chair of technical committee of network and service capability, China Communications Standard Association(CCSA) . His research interests include traffics optimization in SDN/NFV and cloud computing.