Providing VNF Services with Pipe&Hose Model Based Nonblocking SDN Networks

[1]Yangming Zhao, [1]Jingyuan Fan, [2]Huan Chen, [1]Chunming Qiao
[1]Department of Computer Science and Engineering
State University of New York at Buffalo
[2] University of Electronic Science and Technology of China

***Abstract*—Combining Virtual Network Functions (VNF) and Software Defined Networking (SDN) enables fine-grained traffic steering to provide required network services to flows. However, online routing calculation and flow table enforcement incurs an non-negligible overhead. In this work, we propose to design a nonblocking network with fixed routing and VNF provisioning schemes to improve the network performance (e.g. the flow completion time). We first formulate this problem as a linear programming (LP) problem with infinite number of constraints, and leverage primal-dual technology to reformulate the problem as a polynomial size LP. The LP formulation is also extended to reconfigure the network when some components become unavailable, in order to keep the network nonblocking. Since solving the LP formulation is time consuming or even impossible in large scale networks, an efficient algorithm based on optimization decomposition and column generation is proposed to find a near optimal solution quickly. Simulation results show that nonblocking networks can speed up 60% of the flows by 2x, with a small increase in the required network capacity, compared with approaches that do not use the nonblocking networks.**

I. Introduction

In a Virtual Network Functions (VNF) environment, traffic should be steered to pass required middleboxes implementing the required VNFs in a specific order due to the security policy issue. For example, for a secure server whose accesses need to be highly restricted, its incoming traffic may first traverse an FW and then an IDS/IPS. In this paper, we refer such ordered network function chain as a *policy*.

Conventionally, there are two ways to steer the traffic to get its required policy. The first one is to jointly optimize the VNF placement and traffic routing offline [1], while the second is online dynamically steering traffic along the way where it can receive required VNFs by either using the existing middleboxes or provisioning new middleboxes [2]–[4]. For the first method, it assumes the traffic matrix in the network is known or can be measured, which is not a practical assumption. With the increasing popularity of higher bandwidth applications (e.g. file swarming, online and offline video distribution), traffic patterns are more violative even in the aggregate. Further, multi-homing customers cause abrupt changes in aggregate traffic by shifting traffic between networks [5]. The second method may fail to deliver the traffic on time even if there is enough network capacity. For example, in a rearrangeably nonblocking network, there is always enough bandwidth to deliver all the traffic injected into the network, however, it requires to adjust the routing of existing traffic, which may not be allowed. In addition, this method enforces flow entries online, which is not negligible overhead especially for the mice flows [6].

In this paper, we propose to design a *nonblocking* network. Instead of requiring the exact traffic matrix as input, we only assume the traffic matrix lies in a region defined by pipe and hose models. Under the pipe and hose model constraints, we design *fixed* traffic routings and *predefine* where to provide policy for each traffic; for all the possible traffic matrices, we ensure none of the links experience congestion and every node has enough resources to provide policies. The fixed routing and predefined policy providing scheme can be pre-installed in the system, and hence, the overhead for online enforcing the flow entries is eliminated. In addition, a nonblocking network ensures no congestion in the network and hence no queuing delay or packet loss due to the congestion. Accordingly, the network performance can be greatly improved.

There are two challenges to design such a nonblocking network. At first, there are an infinite number of possible traffic matrices satisfying the pipe and hose models, which results in an infinite number of constraints in the optimization problem. More difficultly, these constraints are not only on links but also on the nodes that should prepare enough computing resources to provide policies. In this paper, we leverage the primal-dual technology to solve the infinite number of constraints problem. The other challenge is the problem scalability. Even for Linear Programming (LP) problem, we cannot solve it in a timely manner in large scale networks. In this paper, we propose an efficient algorithm based on optimization decomposition and column generation to address this issue.

The main contributions of this work can be summarized as follows:

- Formulating the problem to design network capacity to form a nonblocking network (Section IV)
- Primal-dual technology to solve the infinite number of constraints problem (Section IV)
- Algorithm based on optimization decomposition and column generation to solve the network reconfiguration problem when some components are unavailable (Section V)

II. Related Work

In this paper, we focus on traffic routing and policy providing scheme. [1] proposes to jointly optimize the middlebox placement and traffic routing, in order to minimize the required

978-1-5386-2542-2/18/$31.00 @2018

amount of resources to satisfy all the policy requirements. [7] is also a work to minimize the required resources but introduces the interference freedom and resource isolation as constraints. These works require a specific traffic matrix as input. Since traffic is time varying, such an input is not always available in practical networks.

[2, 3] are works steering traffic to traverse the required middleboxes. [2] focus on how to control the flow table such that the traffic can traverse the proper middleboxes, while [3] is to calculate the traffic routing such that the traffic throughput can be maximized. In these works, the VNF provided by each node is fixed, while we leverage consolidated middlebox technology to dynamically change the VNF provided by each node according to the traffic.

To design nonblocking networks, there are also a lot of existing works. [8] is a representative among them. It designs the fixed routing to minimize the maximum link utilization by considering both of the pipe and hose models. In addition, [9] is to design nonblocking networks under hose model constraints with the objective of minimizing the total link capacity. All the previous works on the nonblocking network do not consider the computing resources constraints on each node to implement VNFs.

The work most relevant to ours is [4]. In [4], the authors design a scalable routing scheme to provide policies to traffic under the constraints of switch rule capacity and middlebox computation capacity. Similar to our work, [4] assumes the consolidated middlebox is deployed in the network. However, it assumes the exact traffic matrix is previously known.

III. Problem Definition and Notations

A. System Model

In this work, we propose to design a *nonblocking* network, such that none of its internal links will ever experience congestion, and there are enough computing resources to provide policy services to all the traffic. Then, we can fix the traffic routing and predefine which node to fulfill its policy based on its source, destination and policy requirement, although every node can provide any policy as long as sufficient amount of resource is available. By doing so, we can pre-install the flow table to eliminate the online calculation and flow table enforcement. Though the traffic is time varying, we assume the traffic matrix lies in a region defined by pipe and hose models. In addition, container-based technology is leveraged to set up its policy with only several micro-seconds overhead [10], and policies are provided by the consolidated middlebox proposed in [11], i.e. the entire policy of each flow is provided at a single node.

To design a nonblocking network, we focus on following two questions: 1) given the pipe and hose model constraints, how to fix the traffic routing and determine which node to provide policy to each traffic, such that we can minimize the resources to form a nonblocking network; 2) when some network components are unavailable, how to fast recover the services through network reconfiguration.

B. Basic Formulation and Notations

Formally, we consider a network $G = (V, E)$ with node set V and directed link set E. The *link capacity* of $e \in E$ is c_e. $I(u)$ is the set of incoming links to node u, and $O(u)$ is the set of outgoing links from node u. For each node u, it contains different types of computing resources, such as memory, CPU, GPU etc., to realize middleboxes and provide policies. We refer to the amount of computation resources that can be provided by node u as *node capacity*, and use r_{uk} to denote the amount of resource k that can be provided by node u. For simplicity, we use *network capacity* to denote both of the link capacity and the node capacity.

The traffic matrix element t_{ij}^s specifies traffic rate of data from node i to node j with policy requirement s. For simplicity, we also use t_{ij}^s to denote the traffic from node i to node j with policy requirement s hereafter. In the perspective of network operator, t_{ij}^s is the aggregated rate of many flows, and hence it is splittable.

The traffic matrix lies in the region defined by pipe and hose models. Pipe model limits the traffic rate between node pairs:

$$\sum_{s} t_{ij}^s \le w_{ij} \quad \forall i, j \tag{1}$$

where w_{ij} is the traffic limitation from node i to node j; Hose model specifies egress rate upper bound and ingress rate upper bound for each node:

$$\begin{aligned} \sum_{j,s} t_{ij}^s &\le \eta_i \quad \forall i \\ \sum_{i,s} t_{ij}^s &\le \lambda_j \quad \forall j \end{aligned} \tag{2}$$

η_i is the egress upper bound of node i and λ_j is the ingress rate upper bound of node j.

For a valid traffic route, the following *flow conservation constraints* should be satisfied:

$$\sum_{e \in I(u)} f_{ijs}^e - \sum_{e \in O(u)} f_{ijs}^e = \begin{cases} -1 & \text{if } u = i \\ 1 & \text{if } u = j \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j, u, s \tag{3}$$

where f_{ijs}^e is the fraction of traffic t_{ij}^s that is split onto link e. The total traffic on link e is $\sum_{i,j,s} f_{ijs}^e t_{ij}^s$, which should be less than the link capacity, i.e.

$$\sum_{i,j} \sum_{s} f_{ijs}^e t_{ij}^s \le c_e \quad \forall e \tag{4}$$

This is referred as *link capacity constraint*.

The flow conservation constraints can also be presented as *path-based formulation*. If the available paths between any node pair (i, j) for any specific policy s are given, the flow conservation constraints can be formulated as

$$\sum_{p \in P_{ij}^s} x_{ijs}^p = 1 \quad \forall i, j, s \tag{5}$$

where P_{ij}^s is the set of paths that can be used by the traffic t_{ij}^s, and x_{ijs}^p is the fraction of this traffic split onto path p.

Correspondingly, the link capacity constraint is changed to be

$$\sum_{i,j}\sum_{s}\sum_{p:e\in p} x^{p}_{ijs} t^{s}_{ij} \le c_e \quad \forall e \tag{6}$$

Suppose q^{u}_{ijs} is the fraction of traffic t^{s}_{ij} that receives service of policy s at node u, g^{e}_{ijs} and h^{e}_{ijs} are the fraction of traffic t^{s}_{ij} on link e, which has already received service of policy s and has not received service of policy s, respectively, we have:
1) Traffic on each link should be either served or not:

$$g^{e}_{ijs} + h^{e}_{ijs} = f^{e}_{ijs} \quad \forall e \tag{7}$$

2) Unserved traffic should keep conservation

$$\sum_{e\in I(u)} h^{e}_{ijs} - \sum_{e\in O(u)} h^{e}_{ijs} - q^{u}_{ijs} = \begin{cases} -1 & \text{if } u = i \\ 0 & \text{otherwise} \end{cases} \quad \forall i,j,u,s \tag{8}$$

3) All the traffic should receive required policy services

$$\sum_{u} q^{u}_{ijs} = 1 \quad \forall i,j,s \tag{9}$$

Hereafter, we refer above three constraints as *policy providing constraints*. To provide policy services to all the traffic, let b_{sk} be the amount of resource k that is required to serve each unit of traffic with policy s, the following *node capacity constraints* should be satisfied:

$$\sum_{s}\sum_{i,j} q^{u}_{ijs} t^{s}_{ij} b_{sk} \le r_{uk} \quad \forall u,k \tag{10}$$

If the path-based formulation is adopted, let y^{pu}_{ijs} denote the fraction of traffic t^{s}_{ij} carried by path p and receives services of policy s on the node $u \in p$, we have

$$\sum_{u\in p} y^{pu}_{ijs} = x^{p}_{ijs} \quad \forall i,j,s \tag{11}$$

and

$$\sum_{i,j}\sum_{s}\sum_{p\in P^{s}_{ij}:u\in p} y^{pu}_{ijs} t^{s}_{ij} b_{sk} \le r_{uk} \quad \forall u,k \tag{12}$$

(11) says all the traffic on path p should receive required policy service, while (12) indicates there should be enough resources on each node to provide services.

IV. Network Capacity Planning

In this section, we study how to plan the network capacity, including the link capacity c_e and node capacity r_{uk}, such that the network is nonblocking. The objective is to minimize the amount of total resources, i.e.

$$\text{minimize} \qquad \sum_{e} c_e + \sum_{u}\sum_{k} \zeta_k r_{uk} \tag{13}$$

where ζ_k is the weight of resource k relative to the link capacity. Then, combining with necessary constraints, we derive following Network Planning Problem (NPP):

minimize (13)
subject to:
Flow conservation constraints (3)
Link capacity constraint (4)
Policy providing constraints (7) – (9)
Node capacity constraint (10)

It should be noted that there are infinite number of constraints in (4) and (10), which makes the NPP difficult to solve. Fortunately, we have following theorem, which helps set the stage for simplifying the NPP:

Theorem 1. *In the NPP model, the constraint* (4) *can be substituted by*

$$\begin{aligned} &\sum_{i,j} w_{ij}\alpha_{ij}(e) + \sum_{i} \eta_i\beta_i(e) + \sum_{j} \lambda_j\gamma_j(e) \le c_e \\ &\alpha_{ij}(e) + \beta_i(e) + \gamma_j(e) \ge f^{e}_{ijs} \quad \forall i,j,s \\ &\alpha_{ij}(e) \ge 0, \beta_i(e) \ge 0, \gamma_j(e) \ge 0 \end{aligned} \tag{14}$$

for each link e*. In addition, the constraint* (10) *can be reformulated as*

$$\begin{aligned} &\sum_{i,j} w_{ij}\phi_{ij}(u) + \sum_{i} \eta_i\varphi_i(u) + \sum_{j} \lambda_j\psi_j(u) \le r_{uk} \quad \forall k \\ &\phi_{ij}(u) + \varphi_i(u) + \psi_j(u) \ge q^{u}_{ijs} b_{sk} \quad \forall i,j,s \\ &\phi_{ij}(u) \ge 0, \varphi_i(u) \ge 0, \psi_j(u) \ge 0 \end{aligned} \tag{15}$$

for each node u

Due to the space limitation, we omit the proof of this theorem here. Based on Theorem 1, we can derive a polynomial size LP model for NPP.

V. Network Reconfiguration

When some components are unavailable, assume θ_{ij} is the *scale-down ratio*, i.e. serve $\theta_{ij} t^{s}_{ij}$ of t^{s}_{ij} to keep network nonblocking, we are to minimize the amount of traffic reduction, $\sum_{i,j,s}(1-\theta_{ij})t^{s}_{ij}$. Let T denote the maximum amount of traffic that may be reduced, there should be

$$\sum_{i,j}\sum_{s}(1-\theta_{ij})t^{s}_{ij} \le T \tag{16}$$

for all the $\{t^{s}_{ij}\}$ satisfying (1) and (2). Apparently, it can also be replaced by following constraints

$$\begin{aligned} &\sum_{i,j} w_{ij}\kappa_{ij} + \sum_{i} \eta_i\nu_i + \sum_{j} \lambda_j\xi_j \le T \\ &\kappa_{ij} + \nu_i + \xi_j \ge 1 - \theta_{ij} \quad \forall i,j,s \\ &\kappa_{ij} \ge 0, \nu_i \ge 0, \xi_j \ge 0 \end{aligned} \tag{17}$$

Different from TSP model, constraints

$$0 \le \theta_{ij} \le 1 \quad \forall i,j \tag{18}$$

should be added into the formulation, as we cannot enlarge the original traffic request to compensate for the traffic reduction.

To reduce the problem scalability, we present the reconfiguration problem as link-path formulation. Since not all the traffic can be served at the full rate, the (5) should be modified as

$$\sum_{p\in P^{s}_{ij}} x^{p}_{ijs} = \theta_{ij} \tag{19}$$

to indicate that t^{s}_{ij} can be only served at the rate $\theta_{ij} t^{s}_{ij}$. Correspondingly, The equation groups used to substitute (6) and (12) become

$$\begin{aligned}&\sum_{i,j} w_{ij}\alpha_{ij}(e) + \sum_{i} \eta_i\beta_i(e) + \sum_{j} \lambda_j\gamma_j(e) \le c_e\\ &\alpha_{ij}(e) + \beta_i(e) + \gamma_j(e) \ge \sum_{p:e\in p} x^p_{ijs} \quad \forall i,j,s\\ &\alpha_{ij}(e) \ge 0, \beta_i(e) \ge 0, \gamma_j(e) \ge 0\end{aligned} \tag{20}$$

for all link e, and

$$\begin{aligned}&\sum_{i,j} w_{ij}\phi_{ij}(u) + \sum_{i} \eta_i\varphi_i(u) + \sum_{j} \lambda_j\psi_j(u) \le r_{uk} \quad \forall k\\ &\phi_{ij}(u) + \varphi_i(u) + \psi_j(u) \ge b_{sk} \sum_{p:u\in p} y^{pu}_{ijs} \quad \forall i,j,s\\ &\phi_{ij}(u) \ge 0, \varphi_i(u) \ge 0, \psi_j(u) \ge 0\end{aligned} \tag{21}$$

for each node u, respectively. Then, we can formulate the Configuration Update Problem (CUP) as follows:

minimize $\quad T$

subject to:
Path based flow conservation (19)
Link capacity constraint (20)
Node capacity constraint (21)
Policy providing constraint (11)
Traffic loss calculation (17)

CUP includes two subproblems: 1) traffic routing; 2) policy providing. These two subproblems are coupled through constraint (11) and the traffic scale-down ratio θ_{ij}. Accordingly, we treat θ_{ij} as two variables $\theta^{(1)}_{ij}$ and $\theta^{(2)}_{ij}$. $\theta^{(1)}_{ij}$ is associated with the traffic routing subproblem (i.e. constraints (19)), while $\theta^{(2)}_{ij}$ is associated with policy providing problem. To keep the problem equivalent, $\theta^{(2)}_{ij} = \theta^{(1)}_{ij}$ should be added into the formulation. Based on (11), (19) and $\theta^{(2)}_{ij} = \theta^{(1)}_{ij}$, we know

$$\sum_{p\in P^s_{ij}} \sum_{u\in p} y^{pu}_{ijs} = \theta^{(2)}_{ij} \tag{22}$$

Then, by relaxing (11) and $\theta^{(2)}_{ij} = \theta^{(1)}_{ij}$, the CUP can be decomposed into two subproblems:

$$\text{minimize} \quad T + \sum_{i,j}\sum_{s}\sum_{p\in P^s_{ij}} \pi^p_{ijs} x^p_{ijs} + \sum_{ij} \upsilon_{ij}\theta^{(1)}_{ij} \tag{23}$$

$$\text{subject to:} \qquad (17), (19), (20)$$

and

$$\text{minimize} \quad T - \sum_{i,j}\sum_{s}\sum_{p\in P^s_{ij}} \pi^p_{ijs} \sum_{u\in p} y^{pu}_{ijs} - \sum_{ij} \upsilon_{ij}\theta^{(2)}_{ij} \tag{24}$$

$$\text{subject to:} \qquad (17), (21), (22)$$

with the Lagrange multiplier π^p_{ijs} and υ_{ij}. Based on this decomposition, we propose an iterative method to solve CUP. This method includes two steps: 1) solve CUP with given path sets P^s_{ij}; 2) add more paths in the path set to improve the solution.

With given path sets P^s_{ij} and Lagrange multiplier π^p_{ijs} and υ_{ij}, we can solve (23) and (24). Say $\hat{\theta}^{(1)}_{ij}$ and $\hat{x}^p_{ijs}$ are the solutions of (23), and $\hat{\theta}^{(2)}_{ij}$ and $\hat{y}^{pu}_{ijs}$ are the solutions of (24). If $\sum_{u\in p} \hat{y}^{pu}_{ijs} = \hat{x}^p_{ijs}$ does not hold, we update the π^p_{ijs} by $\pi^p_{ijs} \leftarrow \pi^p_{ijs} + \Delta_1(\hat{x}^p_{ijs} - \sum_{u\in p} \hat{y}^{pu}_{ijs})$, where Δ_1 is the step size. Similarly, if $\hat{\theta}^{(2)}_{ij} = \hat{\theta}^{(1)}_{ij}$ does not hold, we update υ_{ij} by $\upsilon_{ij} \leftarrow \upsilon_{ij} + \Delta_2(\hat{\theta}^{(1)}_{ij} - \hat{\theta}^{(2)}_{ij})$, with step size Δ_2. Such procedure ends when both $|\hat{x}^p_{ijs} - \sum_{u\in p} \hat{y}^{pu}_{ijs}|$ and $|\hat{\theta}^{(1)}_{ij} - \hat{\theta}^{(2)}_{ij}|$ are less than a predefined threshold.

To add more paths to improve the objective, the column generation algorithm is leveraged. The variables associated with paths x^p_{ijs} only exist in subproblem (23), hereby we select the path to add based on this subproblem. The reduce cost of adding path p into P^s_{ij} can be calculated as

$$r_p = -\rho_{ij} - \sum_{e\in p} \varsigma_{ijs}(e) \tag{25}$$

where ρ_{ij} and $\varsigma_{ijs}(e)$ are the dual variables of constraints (19) and the second constraints in (20), respectively [8]. Since ρ_{ij} is independent of path p, we can find p^*, such that

$$p^* = \arg\min_{p}\{-\sum_{e\in p} \varsigma_{ijs}(e)\} \tag{26}$$

to add into P^s_{ij}, and add variable $x^{p^*}_{ijs}$ into problem (23). To derive such path, we can find the shortest path between i and j with $-\varsigma_{ijs}(e)$ as the weight of link e. Correspondingly, we add variable $y^{p^*u}_{ijs}$ into problem (24), and go to the first step to solve CUP with the updated path set.

Based on above discussions, we design Algorithm 1 to update the network configurations when some components are unavailable. In this algorithm, Lines 4–8 are used to calculate the best traffic routing and policy providing scheme with the given path set, while Lines 9–13 are used to add more paths to improve the algorithm performance based on column generation.

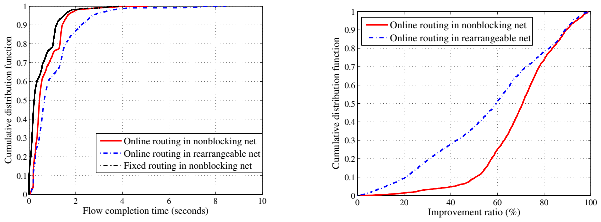
VI. Performance Evaluation

In this section, we use multiple setups to evaluate the performance of our nonblocking system. First, we demonstrate the benefit of building a strictly nonblocking network on Clos topology in Section VI-A. Section VI-B studies the cost to build a nonblocking system on two real Internet topologies. Section VI-C shows the efficiency of the column generation based recovery algorithm.

A. Traffic Performance Improvement

In this section, a packet level simulation is carried out with NS3 [12] on a rearrangeably nonblocking topology, (8,8,8) Clos [13], as it has enough capacity to serve all the traffic in the network if the path can be dynamically changed. The traffic trace is generated based on [14]. We compare the traffic performance under three different settings: 1) design the link capacity and fix the traffic routing to make the network strictly nonblocking by solving NPP with all ingress/egress rate limitation as 10 Gbps; 2) online calculate the traffic

```
Algorithm 1: Network Quick Reconfiguration Algorithm
Input: Initial network configuration $f^e_{ijs}$; unavailable
    network components
Output: New traffic configuration $x^p_{ijs}$, service provision
    configuration $y^{pu}_{ijs}$; traffic shrink ratio $\theta_{ij}$
Initialize path set $P^s_{ij}$ based on input
while Running time is less than the threshold do
    Initialize $\pi^p_{ijs} \leftarrow 0$, $\upsilon_{ij} \leftarrow 0$
    while $|\hat{x}^p_{ijs} - \sum_{u \in p} \hat{y}^{pu}_{ijs}| > \epsilon_1$ or $|\hat{\theta}^{(1)}_{ij} - \hat{\theta}^{(2)}_{ij}| > \epsilon_2$ do
        Solve (23), and get the shrink ratio $\hat{\theta}^{(1)}_{ij}$ and traffic
        routing scheme $\hat{x}^p_{ijs}$
        Solve (24) and yield service provision scheme $\hat{y}^{pu}_{ijs}$
        and traffic shrink ratio $\hat{\theta}^{(2)}_{ij}$
        Update $\pi^p_{ijs}$ by $\pi^p_{ijs} \leftarrow \pi^p_{ijs} + \Delta_1(\hat{x}^p_{ijs} - \sum_{u \in p} \hat{y}^{pu}_{ijs})$,
        $\upsilon_{ij} \leftarrow \upsilon_{ij} + \Delta_2(\hat{\theta}^{(1)}_{ij} - \hat{\theta}^{(2)}_{ij})$
    end while
    for all node pair $(i, j)$ do
        Set weight of link $e$ to be $-\varsigma_{ijs}(e)$
        Find the shortest path $p$ with length $l$
        $P^s_{ij} \leftarrow P^s_{ij} \cup p$
    end for
end while
return $\hat{x}^p_{ijs}$, $\hat{y}^{pu}_{ijs}$ and $\hat{\theta}^{(2)}_{ij}$
```



(a) CDF of flow completion time.

(b) CDF of improvement.

Fig. 1. Flow completion time under different settings.

routing in the strictly nonblocking network designed in 1); 3) online calculate the traffic routing in the (8,8,8) Clos network with all links having the same capacity 10 Gbps by minimizing the maximum link utilization when the traffic arrives in the network. Whenever online routing is employed, we should insert rules into the switches along the selected path. The time to insert one rule into a switch is 18 ms [6].

The CDFs of Flow Completion Time (FCT) under different settings are shown in Fig. 1(a). From this figure, fixed routing can improve the FCT in strictly nonblocking networks as it does not online enforce flow rules into the switch. The FCT in strictly nonblocking networks is less than that in the rearrangeably nonblocking networks. This is due to more capacity allocated in the strictly nonblocking networks and there will be no congestion.

Fig. 1(b) shows the CDF of the FCT improvement ratio, i.e. the percentage of the FCT improvement by the fixed routing in strictly nonblocking networks compared with the FCT in the other two settings. In this figure, we can make two main observations. First, compared with the performance in rearrangeably nonblocking networks that have sufficient capacity to handle all the traffic, 60% of the flows speed up by 2x. By tracing the simulation, we find that for large flows, the main improvement comes from solving the congestion. This leads to relatively smaller improvement ratio since the baseline is large. For mice flows, the improvement approaches 100%, since the time to enforce the flow rules dominates the FCT.

Second, compared with the online routing in nonblocking networks, fixed routing only improves the performance of small flows. As most of the flows in the network are less than 100 KB [15], fixed routing brings large performance improvement (larger than 2x) to most of the flows (more than 90%). Numerically, fixed routing in strictly nonblocking networks brings larger performance improvement compared with online routing in the nonblocking network than that compared with online routing in rearrangeably nonblocking network. It is because there is a smaller FCT baseline in the strictly nonblocking networks.

B. Cost to Build Nonblocking Networks

To build a strictly nonblocking network, we need more resources than the amount that is enough to serve all the traffic. We investigate this cost issue on two real Internet topologies, BT Asia-Pacific and Janet Backbone [16]. We collect traffic data (ingress rate and egress rate of each edge router) from a production network for 2 days and set the maximum ingress/egress rate of each edge router during these 2 days as the hose model constraint. We generate the traffic matrix through gravity model, and the maximum traffic rate between each node pair during the 2 days is treated as the pipe model constraint. For simplicity, we assume there is only one type of computing resources for middleboxes (e.g. CPU), and 10 types of policies in the network. The amount of computing resources required to provide policies to a unit of traffic is randomly distributed on [3,7]. Each flow requires a randomly selected type of policy. For confidentiality reasons, we normalize the traffic rate by the maximum ingress/egress rate among all the routers. The cost weight of computing resource is 0.5, which makes the value of total link capacity be close to the weighted total node capacity.

As a baseline, we route the flows at each time instance one by one with the objective to minimize the maximum link/node utilization and set the largest capacity requirement of each link/node among all time instances as its capacity.

The total amount of network capacity used by different network design methods on BT Asia-Pacific and Janet Backbone are shown in Fig. 2. From this figure, we observe that consider only the hose model, about 10% more link capacity is needed than the online load balance routing scheme in BT Asia-Pacific, while 13% more link capacity is needed in Janet Backbone to design a nonblocking network. This is because that nonblocking network should adapt to all the possible traffic matrices, while the load balance routing scheme only needs to handle the collected ones. When taking the pipe model constraints into consideration, design a nonblocking

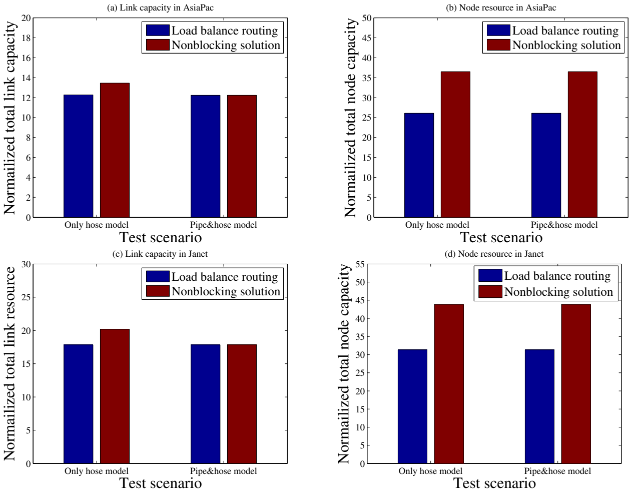
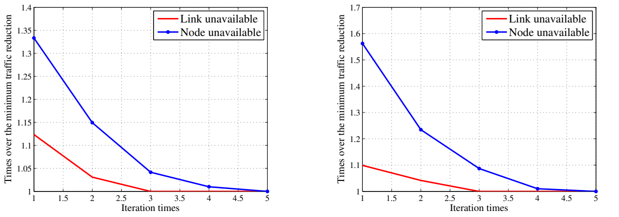


Fig. 2. The cost to design nonblocking networks.



(a) Recovery in BT Asia-Pacific. (b) Recovery in Janet Backbone.

Fig. 3. Incremental recovery performance.

network needs almost the same amount of link capacity since the pipe model provides more detailed traffic information.

For the node capacity, deploy a nonblocking network by solving NPP needs about 1.4x of the capacity required by the load balance routing method. This is because the nonblocking network always considers the worst case, while the load balance routing method considers the average case. As the total node capacity is only determined by the traffic amount injected into the network, rather than its routing, whether or not take the pipe model into consideration does not impact the required node capacity.

C. Incremental Recovery vs. Overall Recovery

When the path-based formulation is leveraged to derive a network reconfiguration scheme quickly, since not all the paths are enumerated, we cannot always obtain the optimal solution. In this section, we study the relationship between the iteration times and the algorithm performance. As a baseline, we directly solve the LP model of the node-link formulation and derive the minimum traffic reduction. In the network designed in Section VI-B, we randomly select one unavailable component and record how the traffic reduction changes with the iteration times. Fig. 3 shows the simulation results.

From this figure, we can make following observations. First, loss of one node needs more iterations to derive a near optimal solution than the loss of one link, since node loss generally breaks more paths. Second, loss of nodes in Janet Backbone incurs relatively more traffic reduction than that in BT Asia-Pacific, while the loss of links in Janet Backbone results in relatively less traffic reduction than in Asia-Pacific. It is because that there are some densely connected nodes in Janet Backbone. The loss of these densely connected nodes results in large traffic reduction and hence increases the average traffic reduction. Due to the same reason, loss of a link in Janet Backbone leads to less damage than in BT Asia-Pacific. Most importantly, the optimal solution can be derived with very few iterations in all the cases. The network configuration yielded by NPP contains most of the paths in the optimal solution, so Algorithm 1 achieves the optimal solution quickly.

VII. Conclusion

In this paper, we proposed to provide VNF services with nonblocking networks. Different from previous works on traffic steering for service chaining, we assume the traffic varies under the constraints of pipe and hose models. Semi-infinite programming models are formulated for network design and network reconfiguration problems. Primal-Dual technology is leveraged to solve the infinite constraint problem, and an efficient algorithm based on optimization decomposition and column generation is proposed to find a near optimal solution quickly. Simulation results show that with nonblocking networks, we can greatly improve the flow completion time at a reasonable resource cost.

References

[1] H. Huang, S. Guo, J. Wu, and J. Li, “Joint middlebox selection and routing for software-defined networking,” in *IEEE ICC*, 2016.

[2] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula, “Steering: A software-defined networking for inline service chaining,” in *IEEE ICNP*, 2013.

[3] L. Guo, J. Pang, and A. Walid, “Dynamic service function chaining in sdn-enabled networks with middleboxes,” in *IEEE ICNP*, 2016.

[4] A. Gushchin, A. Walid, and A. Tang, “Scalable routing in sdn-enabled networks with consolidated middleboxes,” in *Proceedings of the ACM HotMiddleBox*, 2015, pp. 55–60.

[5] F. Guo, J. Chen, W. Li, and T. cker Chiueh, “Experiences in building a multihoming load balancing system,” in *IEEE INFOCOM*, 2004.

[6] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, “Dynamic scheduling of network updates,” in *Proceedings of the ACM SIGCOMM*, 2014, pp. 539–550.

[7] X. Li and C. Qian, “An nfv orchestration framework for interference-free policy enforcement,” in *IEEE ICDCS*, 2016.

[8] V. Tabatabaee, A. Kashyap, B. Bhattacharjee, R. J. La, and M. A. Shayman, “Robust routing with unknown traffic matrices,” in *IEEE INFOCOM*, 2007.

[9] J. Chu and C. T. Lea, “New architecture and algorithms for fast construction of hose-model vpns,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 670–679, June 2008.

[10] R. Cziva, S. Jouet, K. J. S. White, and D. P. Pezaros, “Container-based network function virtualization for software-defined networks,” in *IEEE ISCC*, July 2015, pp. 415–420.

[11] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, “Design and implementation of a consolidated middlebox architecture,” in *Proceedings of the USENIX NSDI 2012*.

[12] “Ns3,” 2016. [Online]. Available: ”https://www.nsnam.org/”

[13] C. Clos, “A study of non-blocking switching networks,” *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[14] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, “The case for evaluating mapreduce performance using workload suites,” in *IEEE MASCOTS*, 2011.

[15] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: Measurements & analysis,” in *Proceedings of the ACM IMC*, 2009.

[16] “Topology zoo,” 2016. [Online]. Available: ”http://www.topology-zoo.org/dataset.html”