On Orchestrating Virtual Network Functions in NFV

Md. Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba
David R. Cheriton School of Computer Science, University of Waterloo
[mfbari | sr2chowdhury | r5ahmed | rboutaba]@uwaterloo.ca

***Abstract*—Middleboxes or network appliances like firewalls, proxies and WAN optimizers have become an integral part of today's ISP and enterprise networks. Middlebox functionalities are usually deployed on expensive and proprietary hardware that require trained personnel for deployment and maintenance. Middleboxes contribute significantly to a network's capital and operational costs. In addition, organizations often require their traffic to pass through a specific sequence of middleboxes for compliance with security and performance policies. This makes the middlebox deployment and maintenance tasks even more complicated. Network Function Virtualization (NFV) is an emerging and promising technology that is envisioned to overcome these challenges. It proposes to move packet processing from dedicated hardware middleboxes to software running on commodity servers. In NFV terminology, software middleboxes are referred to as Virtualized Network Functions (VNFs). It is a challenging problem to determine the required number and placement of VNFs that optimizes network operational costs and utilization, without violating service level agreements. We call this the VNF Orchestration Problem (VNF-OP) and provide an Integer Linear Programming (ILP) formulation with implementation in CPLEX. We also provide a dynamic programming based heuristic to solve larger instances of VNF-OP. Trace driven simulations on real-world network topologies demonstrate that the heuristic can provide solutions that are within 1.3 times of the optimal solution. Our experiments suggest that a VNF based approach can provide more than $4\times$ reduction in the operational cost of a network.**

I. Introduction

Today's enterprise networks ubiquitously deploy vertically integrated proprietary middleboxes or network appliances to offer various network services. Examples of such middleboxes include firewalls, proxies, WAN optimizers, Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs). A recent study shows that the number of different middleboxes is comparable to the number of routers in an enterprise network [33]. However, middleboxes come with high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX). They are usually expensive, vendor specific and require specially trained personnel for deployment and maintenance. Moreover, it is often impossible to add new functionality to an existing middlebox, which makes it very difficult for the network operators to deploy new services. In many cases, network operators are compelled to purchase new hardware that substantially increases their CAPEX.

Another set of problems arise from the fact that most often a traffic is required to pass through multiple stages of middlebox processing in a particular order, *e.g.*, a traffic may be required to go through an IDS, then a proxy and finally through a firewall [27]. This phenomenon is very common for middleboxes and is typically referred to as Service Function Chaining (SFC) [28]. The IETF Network and Service Chaining Working Group has several IETF drafts demonstrating middlebox chaining use-cases in operator networks [21], mobile networks [17] and data center networks [35]. The task of sequencing these in-network processing is commonly referred to as *middlebox orchestration*. Currently, this task is performed by crafting the routing table entries manually. It is a cumbersome and error-prone process. Moreover, any placement of these hardware middleboxes is bound to become inefficient over time. It is very expensive and inconvenient to keep changing the locations of these hardware middleboxes with changing network conditions.

An emerging and promising technology that can address these limitations is Network Function Virtualization (NFV) [11]. It proposes to move packet processing from hardware middleboxes to software. Instead of running hardware based middleboxes, the same packet processing tasks are performed by software middleboxes running on commodity (*e.g.*, `x86` based systems) servers. Researchers have already developed virtualized platforms for software based middlebox processing that can achieve near hardware performance [18], [22]. In NFV terminology, these software middleboxes are referred to as Virtualized Network Functions (VNFs). NFV is envisioned to solve most of the above mentioned problems with hardware middleboxes. NFV also provides opportunities for network optimization and cost reduction. Previously, middleboxes were hardware appliances placed at fixed locations, but we can deploy a VNF virtually anywhere in the network. This feature opens-up a whole new window of opportunity to reduce both CAPEX and OPEX. Network operators no longer need to buy specialized hardware. This would significantly reduce CAPEX. In addition, VNFs can be orchestrated autonomically without requiring specially trained personnel for deployment and maintenance, which would reduce the operational and maintenance costs.

VNFs can be orchestrated by deploying a composition of VNFs either on the same server or on a cluster of servers. The locations of the VNFs must be chosen carefully to ensure that traffic can be routed through them in the proper sequence with minimal changes in the forwarding tables. An emerging technology that can assist in flexible routing is Software Defined Networking (SDN) [20]. SDN decouples the control plane from the data plane and places it on a logically centralized controller. SDN control plane has a global network view and can be used to programmatically configure forwarding rules in the switches/routers to enable VNF orchestration.

VNFs promise to reduce the CAPEX and OPEX of a

network. However, several issues need to be considered before provisioning VNFs: (i) the cost of deploying a new VNF, (ii) energy cost for running a VNF and (iii) the cost of forwarding traffic to and from a VNF. Placing just enough VNFs to match processing requirements of traffic may yield the lowest deployment and energy cost, but steering traffic through these VNFs will result in the use of sub-optimal paths that may lead to Service Level Objective (SLO) violations and customer dissatisfaction. On the other hand, one may try to always forward traffic through the shortest possible path by deploying VNFs whenever needed. This approach may avoid SLO violation penalty, but will surely lead to huge deployment and energy cost. An optimal VNF placement strategy is needed to find a suitable point between these two extreme cases, *i.e.*, a strategy that minimizes OPEX, penalty for SLO violations and resource fragmentation as well as forwards traffic through the best available path. We refer to this problem as the *Virtualized Network Function Orchestration Problem (VNF-OP)*.

Our key contributions can be summarized as follows:

- We identify the VNF orchestration problem and provide the first quantifiable results showing that dynamic VNF orchestration can have more than $4\times$ reduction in OPEX.
- The problem is formulated as an Integer Liner Program (ILP) and implemented in CPLEX to find optimal solutions for small scale networks.
- Finally, we propose a fast heuristic algorithm that can find solutions within 1.3 times of the optimal. Its performance is evaluated using real-world topologies and traffic traces.

The rest of the paper is organized as follows: we start by explaining the mathematical model used for our system and by formally defining the VNF Orchestration Problem (Section II). Then the problem formulation is presented (Section III). Next, a heuristic is proposed to obtain near-optimal solutions (Section IV). We validate our solution through trace driven simulations on real-world network topologies (Section V). Then, we provide a literature review (Section VI) and finally, we conclude with some future directions (Section VII).

II. Mathematical Model and Problem Definition

In this section we introduce the mathematical model for our system and formally define the VNF Orchestration Problem.

A. Physical Network

We represent the physical network as an undirected graph $\bar{G} = (\bar{S}, \bar{L})$, where $\bar{S}$ and $\bar{L}$ denote the set of switches and links, respectively. We assume that VNFs can be deployed on commodity servers located within the network. The set $\bar{N}$ represents these servers and the binary variable $\bar{h}_{\bar{n}\bar{s}} \in \{0, 1\}$ indicates whether server $\bar{n} \in \bar{N}$ is attached to switch $\bar{s} \in \bar{S}$.

$$\bar{h}_{\bar{n}\bar{s}} = \begin{cases} 1 & \text{if server } \bar{n} \in \bar{N} \text{ is attached to switch } \bar{s} \in \bar{S}, \\ 0 & \text{otherwise.} \end{cases}$$

Let, R denote the set of resources (CPU, memory, disk, *etc.*) offered by each server. The resource capacity of server $\bar{n}$ is denoted by $c^r_{\bar{n}} \in \mathbb{R}^+, \ \forall\ r \in R$. The bandwidth capacity and propagation delay of a physical link $(\bar{u}, \bar{v}) \in \bar{L}$ is represented by $\beta_{\bar{u}\bar{v}} \in \mathbb{R}^+$ and $\delta_{\bar{u}\bar{v}} \in \mathbb{R}^+$, respectively. We also define a function $\eta(\bar{u})$ that returns the neighbors of switch $\bar{u}$.

$$\eta(\bar{u}) = \{\bar{v} \mid (\bar{u}, \bar{v}) \in \bar{L} \text{ or } (\bar{v}, \bar{u}) \in \bar{L}\}, \ \bar{u}, \bar{v} \in \bar{S}$$

B. Virtualized Network Functions (VNFs)

Different types of VNFs (*e.g.*, firewall, IDS, IPS, proxy, *etc.*) can be provisioned in a network. Set P represents the possible VNF types. VNF type p has a specific deployment cost, resource requirement, processing capacity and processing delay represented by $\mathcal{D}^+_p$, $\kappa^r_p \in \mathbb{R}^+(\forall r \in R)$, c_p (in Mbps) and δ_p (in ms), respectively.

There can be certain hardware requirements (*e.g.*, hardware-accelerated encryption for Deep Packet Inspection (DPI)) that may prevent a server from running a particular type of VNF. Furthermore, the network manager may have preferences regarding provisioning a particular type of VNF on a particular set of servers, *e.g.*, Firewalls should be run close to the edge of the network. So, we assume that for each VNF type there is a set of servers on which it can be provisioned. The following binary variable represents this relationship:

$$d_{\bar{n}p} = \begin{cases} 1 & \text{if VNF type } p \in P \text{ can be provisioned on } \bar{n}, \\ 0 & \text{otherwise.} \end{cases}$$

C. Traffic Request

We assume that the network operator is receiving requests for setting up paths for different kinds of traffic (*e.g.*, VPN setup, expected traffic for a new application or service in a data center, *etc.*). A traffic request is represented by a 6-tuple $t = \langle \bar{u}^t, \bar{v}^t, \Psi^t, \beta^t, \delta^t, \omega^t \rangle$, where $\bar{u}^t, \bar{v}^t \in \bar{S}$ denote the ingress and egress switches, respectively. $\beta^t \in \mathbb{R}^+$ is the bandwidth demand of the traffic. δ^t is the expected propagation delay according to Service Level Agreement (SLA). Ψ^t represents the ordered VNF sequence the traffic must pass through (*e.g.*, Firewall $\rightarrowtail$ IDS $\rightarrowtail$ Proxy). l_{Ψ^t} denotes the length of Ψ^t and ω^t denotes policy to determine SLO violation penalties.

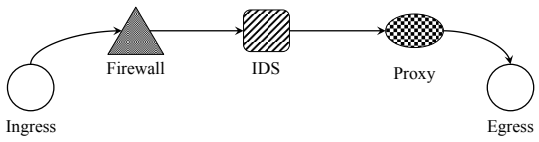


Fig. 1. Traffic Model

We represent a traffic request t by a directed graph $G^t = (N^t, L^t)$, where N^t represents the set of *traffic nodes* (switches and VNFs) and L^t denotes the links between them. Fig. 1 represents a traffic that requires to pass through the VNF sequence: Firewall $\rightarrowtail$ IDS $\rightarrowtail$ Proxy. Modeling the traffic in this way makes it easy for the provisioning process to ensure that it passes though the correct sequence of VNFs. We also define $\eta^t(n_1)$ to represent the neighbors of $n_1 \in N^t$:

$$\eta^t(n_1) = \{n_2 \mid (n_1, n_2) \in L^t\},\ n_1, n_2 \in N^t$$

Next, we define a binary variable $g^t_{np} \in \{0, 1\}$ to indicate the type of a node $n \in N^t$

$$g^t_{np} = \begin{cases} 1 & \text{if node } n \in N^t \text{ is of type } p \in P, \\ 0 & \text{otherwise.} \end{cases}$$

D. VNF Orchestration Problem (VNF-OP)

We consider a scenario where an operational network is serving a set of traffics $\hat{T}$. It has a set of VNFs already deployed and the routing paths for the traffics in $\hat{T}$ are also provisioned. Now, the network operator is receiving new traffic requests and wants to provision the required VNFs and routing paths for them. The network operator can choose to provision resources for one traffic request at a time or leverage a lookahead interval by accumulating a number of traffic requests and provision resources in batches. Determining the optimal number or volume of traffic or the length of the lookahead interval for each batch is an interesting research challenge that we consider out-of-scope for the current work and plan to pursue in the future. In the rest of the paper, we denote a new traffic batch by T. Based on the operator's choice, a batch may contain just one or multiple traffic requests.

In the VNF-OP, we are given a physical network topology, VNF specifications, current network status and a set of new traffic requests. Our objective is to minimize the overall network OPEX and physical resource fragmentation by (i) provisioning an optimal number of VNFs, (ii) placing them at the optimal locations and (ii) finding the optimal routing paths for each traffic request, while respecting the capacity constraints (*e.g.*, physical servers, links, and VNFs) and ensuring that traffic passes through the proper VNF sequence.

– OPEX: In this work, we consider the network OPEX to be composed of the following four cost components:

- *VNF deployment cost:* we need to complete tasks like transferring a VM image, booting it and attaching it to devices before deploying a VNF. We associate a cost (in dollars) with these operations.
- Energy cost: it represents the cost of energy consumption for the active servers. A server is considered active if it has at least one active VNF. Servers consume power based on the amount of resources (*e.g.*, CPU, memory, disk, *etc*.) under use. A server is assumed to be in the idle state if it does not have any active VNFs [1].
- Traffic forwarding cost: an operator incurs traffic forwarding cost from two sources: (i) leasing cost of transit links [4] and (ii) energy consumption of the network devices (*e.g.*, switches, routers, *etc*.).
- Penalty for SLO violation: this cost component represents the penalty that must be paid to the customer for SLO violations, *e.g.*, if a traffic experienced more that the maximum allowed propagation delay.

– Resource Fragmentation: We compute the physical resource fragmentation by measuring the percentage of idle resources for the active servers and links. We want to minimize fragmentation as it eventually increases the possibility of accommodating more traffic on the same resource.

III. Integer Linear Programming (ILP) Formulation

VNF-OP is a considerably harder problem to solve than traditional Virtual Network (VN) embedding problems [10]. There is no node ordering requirement in VN embedding, while in VNF-OP we need to preserve the ordering of VNFs. Moreover, in VNF-OP we need to respect the processing capacity constraints of servers and the VNFs to be deployed. How many VNFs are to be deployed is not known in advance, rather it is an outcome of the optimization process. Multi-dimensional Bin Packing [19] can also be used to solve VNF-OP, but here we will end-up with a *nested* bin packing problem. In the first layer traffics need to be packed into VNFs and in the next layer VNFs need to be packed into the physical servers. The fact that the number and locations of VNFs is not known in advance, results in quadratic constraints for resource capacity and renders the problem unsolvable even for very small instances by existing optimization solvers. In this work, we address these challenges by judiciously augmenting the physical network, which is explained in the rest of the section.

A. Physical Network Transformation

We transform the physical network to generate an augmented pseudo-network that reduces the complexity involved in solving the VNF-OP. The transformation process is performed in two steps:

1) VNF Enumeration: A part of the original physical network topology is shown in Fig. 2(a). Here, we have three switches ($s1, s2$ and $s3$) and a server $n2$ connected to switch $s2$. The first transformation is called VNF enumeration, as we enumerate all possible VNFs in this step. The modified network after the first transformation is shown in Fig. 2(b). In this step, we find the maximum number for each type of VNF that can be deployed on each server. We calculate this number based on the resource capacity of the server and the resource requirement of a type of VNF. For example, if a server has 16 cores, and CPU requirement for Firewall and IDS are 4 and 8 cores, respectively, we can deploy 4 Firewalls and 2 IDSs on it. In Fig. 2(b) we show enumerated VNFs for server $n2$.

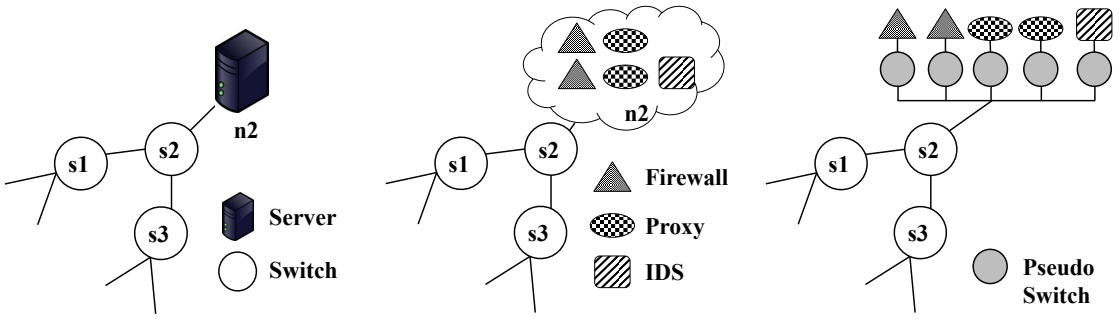
We denote the set of these VNFs (called pseudo-VNFs) by $\mathcal{M}$. Each VNF $m \in \mathcal{M}$ is implicitly attached to a server $\bar{n} \in \bar{N}$. We use the function $\zeta(m)$ to denote this mapping.

$$\zeta(m) = \bar{n} \text{ if VNF } m \text{ is attached to server } \bar{n}$$

We also define a function $\Omega(\bar{n})$ to represent this mapping in the opposite direction:

$$\Omega(\bar{n}) = \{m \mid \zeta(m) = \bar{n}\},\ m \in \mathcal{M}, \bar{n} \in \bar{N}$$

Next, we define $q_{mp} \in \{0, 1\}$ to indicate the type of a VNF:



(a) Original Network (b) VNF Enumeration (c) Adding pseudo-switches

Fig. 2. Network Transformation

$$q_{mp} = \begin{cases} 1 & \text{if VNF } m \text{ is of type } p \in P, \\ 0 & \text{otherwise.} \end{cases}$$

As discussed earlier, a given type of VNF can be deployed on a specific set of servers. To ensure this we must have:

$$q_{mp} = d_{\zeta(m)p} \tag{1}$$

We should note that pseudo-VNFs simply represent where a particular type of VNF can be provisioned. $y_m \in \{0, 1\}$ indicates whether a pseudo-VNF is active or not.

$$y_m = \begin{cases} 1 & \text{if pseudo-VNF } m \in \mathcal{M} \text{ is active,} \\ 0 & \text{otherwise.} \end{cases}$$

2) Adding Pseudo-Switches: Next, we augment the physical topology again by adding a pseudo-switch between each pseudo-VNF and the original switch to which it was connected. This process is shown in Fig. 2(c). We perform this step to simplify the expressions of the network flow conservation constraint in the ILP formulation presented next. This process does not increase the size of the solution space as we consider them only for the flow conservation constraint.

B. ILP Formulation

We define the decision variable x^t_{nm} to represent the mapping of a traffic node to a pseudo-VNF:

$$x^t_{nm} = \begin{cases} 1 & \text{if node } n \in N^t \text{ is provisioned on } m \in \mathcal{M}, \\ 0 & \text{otherwise.} \end{cases}$$

Next, we define another variable to represent the mapping between a traffic node and a switch in the physical network.

$$z^t_{n\bar{s}} = \begin{cases} 1 & \text{if node } n \in N^t \text{ is attached to switch } \bar{s}, \\ 0 & \text{otherwise.} \end{cases}$$

$z^t_{n\bar{s}}$ is not a decision variable as it can be derived from x^t_{nm}:

$$z^t_{n\bar{s}} = 1 \text{ if } x^t_{nm} = 1 \text{ and } \bar{h}_{\zeta(m)\bar{s}} = 1$$

We can also derive the variable y_m from x^t_{nm} as follows:

$$y_m = 1 \text{ if } \sum_{t \in T} \sum_{n \in N^t} x^t_{nm} > 0$$

We assume that $\hat{x}^t_{nm}$ represents the value of x^t_{nm} at the last traffic provisioning event. To ensure that resources for previously provisioned traffic are not deallocated we must have $x^t_{nm} \geq \hat{x}^t_{nm},\ \forall\ t \in \hat{T}, n \in N^t, m \in \mathcal{M}$. Now, we define $\hat{y}_m \in \{0, 1\}$ that represents the value of y_m at the last traffic provisioning event as follows:

$$\hat{y}_m = 1 \text{ if } \sum_{t \in T} \sum_{n \in N^t} \hat{x}^t_{nm} > 0$$

Again, to ensure that resources for previously provisioned traffics are not deallocated we must have $y_m \geq \hat{y}_m,\ \forall\ m \in \mathcal{M}$. Next, we need to ensure that VNF capacities are not overcommitted. The processing capacity of an active VNF must be greater than or equal to the total amount of traffic passing through it. We express this constraint as follows:

$$\sum_{t \in T} \sum_{n \in N^t} x^t_{nm} \times \beta^t \leq c_m,\ \forall\ m \in \mathcal{M} | y_m = 1 \tag{2}$$

We also need to make sure that physical server capacity constraints are not violated by the deployed VNFs. We represent this constraint as follows:

$$\sum_{m \in \Omega(\bar{n})} y_m \times \kappa^r_m \leq c^r_{\bar{n}},\ \forall\ \bar{n} \in \bar{N}, r \in R \tag{3}$$

Each node of a traffic must be mapped to a proper VNF type. This constraint is represented as follows:

$$x^t_{nm} \times g^t_{np} = q_{mp},\ \forall\ t \in T, n \in N^t, m \in \mathcal{M}, p \in P \tag{4}$$

Next, we need to ensure that every traffic node is provisioned and to exactly one VNF.

$$\sum_{t\in T}\sum_{n\in N^t} x^t_{nm} = 1,\ \forall\ m \in \mathcal{M} \tag{5}$$

Now, we define our second decision variable to represent the mapping between links in the traffic model (Fig. 1) to the links in the physical network.

$$w^{tn_1n_2}_{\bar{u}\bar{v}} = \begin{cases} 1 & \text{if } (n1, n2) \in L^t \text{ uses physical link } (\bar{u}, \bar{v}), \\ 0 & \text{otherwise.} \end{cases}$$

We also assume that $\hat{w}^{tn_1n_2}_{\bar{u}\bar{v}}$ represents the value of $w^{tn_1n_2}_{\bar{u}\bar{v}}$ at the last traffic provisioning event. To ensure that resources for previously provisioned traffics are not deallocated in the current iteration we must have

$$w^{tn_1n_2}_{\bar{u}\bar{v}} \geq \hat{w}^{tn_1n_2}_{\bar{u}\bar{v}}, \forall\ t \in \hat{T}, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1, \bar{u}, \bar{v} \in \bar{S} \tag{6}$$

To ensure that each directed link in a traffic request is not mapped to both directions of a physical link, we must have:

$$w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}} \leq 1, \forall\ t \in T, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1, \bar{u}, \bar{v} \in \bar{S} \tag{7}$$

Now, we present the capacity constraint for physical links:

$$\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\bar{S}} (w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}}) \times \beta^t \leq \beta_{\bar{u}\bar{u}}, \forall\ t \in T, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1 \tag{8}$$

Next, we present the flow constraint that makes sure that the in-flow and out-flow of each switch in the physical network is equal except at the ingress and egress switches:

$$\sum_{\bar{v}\in\eta(\bar{u})} \left(w^{tn_1n_2}_{\bar{u}\bar{v}} - w^{tn_1n_2}_{\bar{v}\bar{u}} \right) = z^t_{n_1\bar{u}} - z^t_{n_2\bar{u}}, \forall\ t \in T, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1, \bar{u} \in \bar{S} \tag{9}$$

Finally, we need to ensure that every link in a traffic request is provisioned on one or more physical links in the network:

$$\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\bar{S}} (w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}}) \geq 0, \forall\ t \in T, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1 \tag{10}$$

Our objective is to find the optimal number and placement of VNFs that minimizes OPEX and physical resource fragmentation in the network. We formulate them in detail below:

– OPEX: We consider four cost components to contribute to OPEX. These are as follows:

1. VNF Deployment Cost: the VNF deployment cost can be expressed as follows:

$$\mathbb{D} = \sum_{m\in\mathcal{M}|y_m=1} \mathcal{D}^+_p \times q_{mp} \times (y_m - \hat{y}_m) \tag{11}$$

2. Energy Cost: Without loss of generality we assume that the energy consumption of a server is proportional to the amount of resources being used. However, a server usually consumes power even in the idle state. So, we compute the power consumption of a server as follows:

$$\mathbb{E}_{\bar{n}} = \sum_{m\in\Omega\bar{n}} y_m \times q_{mp} \times e^r(c^r_{\bar{n}}, \kappa^r_p)$$

where

$$e^r(r_t, r_c) = (e^r_{max} - e^r_{idle}) \times \frac{r_c}{r_t} + e^r_{idle}$$

Here, r_t and r_c denote the total and consumed resource, respectively. e^r_{idle} and e^r_{max} denote the energy cost in the idle and peak consumption state for resource r, respectively.

Now, the total energy cost is

$$\mathbb{E} = \sum_{\bar{n}\in\bar{N}}\sum_{m\in\Omega\bar{n}} y_m \times q_{mp} \times e^r(c^r_{\bar{n}}, \kappa^r_p) \tag{12}$$

3. Cost of Forwarding Traffic: Let us assume that the cost of forwarding 1 Mbit data through one link in the network is σ (in dollars). Now, we can compute the total cost of traffic forwarding as follows:

$$\mathbb{F} = \sum_{t\in T}\sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1) \\ \text{and } n_2>n_1}}\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})} \Big((w^{tn_1n_2}_{\bar{u}\bar{v}} - \hat{w}^{tn_1n_2}_{\bar{u}\bar{v}}) \times \beta^t \times \sigma \Big) \tag{13}$$

4. Penalty for SLO violation: We can compute the actual propagation delay experienced by a traffic as follows:

$$\delta^a_t = \sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1) \\ \text{and } n_2>n_1}}\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})} w^{tn_1n_2}_{\bar{u}\bar{v}}\delta_{\bar{u}\bar{v}}$$

Let $\rho^t(\omega^t, \delta^t, \delta^t_a)$ be a function that computes the penalty for SLO violation given the policy for determining penalty (ω^t), expected propagation delay (δ^t) and actual propagation delay (δ^t_a) for traffic t. So, the total cost for SLO violations can be expressed as follows:

$$\mathbb{P} = \sum_{t\in T} \rho^t(\omega^t, \delta_t, \delta^t_a) \tag{14}$$

– Resource Fragmentation: Our second objective is to minimize resource (*e.g.*, server and links) fragmentation. We represent it in terms of dollar as we do for the above mentioned costs. For this purpose, we assume that p^r denotes the price of unit resource of type $r \in R$. We also denote p^β as the price

of unit bandwidth. Now, we can compute the total cost for resource fragmentation as follows:

$$\mathbb{C} = \sum_{\bar{n}\in\bar{N}}\sum_{r\in R}\Big(c^r_{\bar{n}} - \sum_{m\in\omega(\bar{n})}(\kappa^r_p \times q_{mp} \times y_m)\Big) \times p^r + \sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})}\Big(\beta_{\bar{u}\bar{v}} - \sum_{t\in T}\sum_{n_1\in N^t}\sum_{\substack{n_2\in\eta^t(n_1)\\ \text{and } n_2>n_1}}(w^{tn_1n_2}_{\bar{u}\bar{v}} \times \beta^t)\Big) \times p^\beta$$

Here, the first term represents the cost of server resource fragmentation (*e.g.*, CPU, memory, disk, *etc*.) and the second term represents the cost of link bandwidth fragmentation.

Our objective is to minimize the total network operational cost and resource fragmentation that can be expressed as a weighted sum of the aforementioned costs.

$$\text{minimize } \Big(\alpha\mathbb{D} + \beta\mathbb{E} + \gamma\mathbb{F} + \lambda\mathbb{P} + \mu\mathbb{C}\Big) \tag{15}$$

Here, α, β, γ, λ and μ are weighting factors that are used to adjust the relative importance of the cost components.

The VNF-OP is NP-Hard as we can reduce this problem to the *Multi-Commodity, Multi-Plant, Capacitated Facility Location Problem* [26] or more commonly known as the *Trans-shipment Problem* [9] by imposing a constraint on the maximum number of VNFs that can be deployed in the network. Both of these problems are known to be NP-Hard. So, VNF-OP is NP-Hard as well. Therefore in the next section we propose a heuristic to solve this problem.

IV. Heuristic Solution

In this section, we present a heuristic to solve the VNF-OP. Given a network topology, a set of middlebox specifications and a batch of traffic requests, the heuristic finds the number and locations of different types of VNFs required to operate the network with minimal OPEX. We did not explicitly consider resource fragmentation to keep the heuristic simple and fast. However, our experimental results show that even with this simplification, the heuristic produces solutions that are very close to the optimal. The heuristic runs in two steps. First, we model the VNF-OP as a multi-stage directed graph with associated costs. Then we find a near-optimal VNF placement from the multi-stage graph by running the Viterbi algorithm [13]. In the following, we first describe the modeling of VNF-OP using multi-stage graph (Section IV-A), followed by the solution using Viterbi algorithm (Section IV-B). A detailed discussion of the heuristic along with an illustrative example is provided in the Appendix.

A. Modeling with Multi-Stage Graph

For a given traffic request, $t = \langle \bar{u}^t, \bar{v}^t, \Psi^t, \beta^t, \delta^t, \omega^t\rangle$, we represent t as a multi-stage graph with $l_{\Psi^t} + 2$ stages. The first and the last $(l_{\Psi^t} + 2)$ stages represent the ingress and egresses switches, respectively. These two stages contain only one node representing $\bar{u}^t$ and $\bar{v}^t$, respectively. Stage i $(\forall i \in \{2, \dots (l_{\Psi^t}+1)\})$, represents the $(i-1)$-th VNF and the node(s) within this stage represent the possible server locations where a VNF can be placed. Each node is associated with a VNF deployment cost (Eq. 11) and an energy cost (Eq. 12) as described in Section III-B.

An edge $(\bar{v}_i, \bar{v}_j)$ in this multi-stage graph represents the placement of a VNF at a server attached to switch $\bar{v}_j$, given that the previous VNF in the sequence is deployed on a server attached to switch $\bar{v}_i$. We put a directed edge between all pairs of nodes in stage i and $i+1$ $(\forall i \in \{1, 2, \dots (l_{\Psi^t}+1)\})$. We associate two costs with each edge: the cost for forwarding traffic (Eq. 13) and the penalty for SLO violations (Eq. 14). The traffic forwarding cost is proportional to the weighted shortest path (in terms of latency) between the switches. The penalty for SLO violations is obtained by the following process: (i) we equally divide the maximum allowed delay between the stages, (ii) we assign a SLO violation cost for a transition between two successive stages in the multi-stage graph whenever we incur more than the allocated delay due to traffic transport and processing at the nodes. The total cost of a transition between two successive stages in the multi-stage graph is calculated by summing the node and edge costs following Eq. 15. Finally, a path from the node in the first stage to the node in the last stage represents a placement of the VNFs. Our goal is to find a path in the multi-stage graph that yields minimal OPEX.

B. Finding a Near-Optimal Solution

Viterbi algorithm a widely used method for finding the most likely sequence of states from a set of observed states. To find such a sequence, Viterbi algorithm first models the states and their relationships as a multi-stage graph. Each stage consists of the possible states and a transition cost is assigned between all pairs of states in successive stages. Once the multi-stage graph is constructed, Viterbi algorithm proceeds by computing a per node cumulative cost, $cost_u$, recursively defined as the minimum of $cost_v + transition\_cost(v, u)$, for all v in the previous stage as of u's stage. $cost_u$ represents the cost of including node u in the final solution. This computation

Algorithm 1 ProvisionTraffic(t, $\bar{G}$)

$\forall (i,j) \in \{1 \dots |\Psi^t|\} \times \{1 \dots |\bar{S}|\} : cost_{i,j} \leftarrow \infty, \pi_{i,j} \leftarrow NIL$
$\forall i \in |\bar{S}|$:
if $IsResourceAvailable(u^t, i, \Psi^t_1, t)$ **then**
  $cost_{1,n} \leftarrow GetCost(u^t, i, \Psi^t_1, t)$, $\pi_{1,n} \leftarrow n$
end if
$\forall (i,j,k) \in \{2 \dots |\Psi^t|\} \times \{1 \dots |\bar{S}| \times \{1 \dots |\bar{S}|\}$:
if $IsResourceAvailable(k, j, \Psi^t_i, t)$ **then**
  $cost_{i,j} \leftarrow \min\{cost_{i,j}, cost_{i-1,k} + GetCost(k, j, \Psi^t_i, t)\}$
  $\pi_{i,j} \leftarrow i$ yielding minimum $cost_{i,j}$
end if
$\Pi \leftarrow NIL$, $C \leftarrow \infty$, $\psi \leftarrow <>$
$\forall i \in |\bar{S}|$:
  $C \leftarrow \min\{C, cost_{|\Psi^t|,i} + ForwardingCost(i, v^t) + SLOViolationCost(i, v^t, t)\}$
  $\Pi \leftarrow i$ yielding minimum $cost_{|\Psi^t|,i}$
$\forall i \in < |\Psi^t|, |\Psi^t| - 1 \dots 1 >$: Append Π to ψ, $\Pi \leftarrow \pi_{i,\Pi}$
return $Reverse(\psi)$

proceeds in the increasing order of stages. After finishing the computation at the final stage, the most likely sequence of states is constructed by tracing back a path from the final stage back to the first that yields the minimum cost.

We borrow the idea of how costs are computed from Viterbi Algorithm and propose a traffic provisioning algorithm, $ProvisionTraffic$ (Algorithm 1). It takes a traffic request t and a network topology $\bar{G}$ as input and returns a placement of Ψ^t in $\bar{G}$. For each node u in each stage i, we find a node v in stage $i-1$ that yields the minimum total cost $cost_{v,u}$ (costs are defined according to the discussion in Section IV-A). We keep track of the minimum cost path using the table π. After finishing computation for the final stage, we construct the desired VNF placement by back tracing pointers from the final stage of the multi-stage graph to the first stage, using the entries in π. During this process we update residual resource capacities of the servers and the residual bandwidth of the links after each path is allocated. For each traffic request the heuristic solution runs in $\Theta(n^2m)$ time, where n is the number of switches in the network and m is the VNF sequence length (See Appendix B for further details).

V. Performance Evaluation

We perform trace driven simulations on real-world network topologies to gain a deeper insight, and to evaluate the effectiveness of the proposed solution. Our simulation is focused on the following aspects: (i) demonstrating the benefits of dynamic VNF orchestration over hardware middleboxes (Section V-C), (ii) comparing the performance of the heuristic solution with that of the CPLEX based optimal solution (Section V-D) and (iii) Analyzing the behavior of the proposed solution for different traffic volume (Section V-E). Before presenting the results, we briefly describe the simulation setup (Section V-A) and the evaluation metrics (Section V-B). Implementations of both CPLEX and heuristic are available at `http://goo.gl/Da7EZu`.

A. Simulation Setup

1) Topology Dataset: We have used a wide range of network topologies: (i) Internet2 research network (12 nodes, 15 links) [5], (ii) A university data center network (23 nodes, 42 links) [8] and (iii) Autonomous System 3967 (AS-3967) from Rocketfuel topology dataset (79 nodes, 147 links) [34].

2) Traffic Dataset: We use both real traces and synthetically generated traffic for the evaluation. We use traffic matrix traces from [5] to generate time varying traffic for the Internet2 topology. This trace contains a snapshot of a 12×12 traffic matrix and demonstrates significant variability in traffic volume. For the data center network, we use the traces available from [8], and replay the traffic between random source-destination pairs. Finally, for the Rocketfuel topology, we generated a synthetic time-varying traffic matrix using the FNSS tool [30]. It follows the distribution from [25] and exhibits time-of-day effect.

3) Middlebox and Cost Data: We have generated a 3-length middlebox sequence for each traffic based on the data provided in [2], [27]. We have used publicly available data sheets from manufacturers and service providers to select and infer values for server energy cost, SLO violation cost (for violating maximum latency), resource requirements for software middleboxes and their processing capacities. We also obtained energy consumption data for hardware middleboxes from a popular network equipment manufacturer. Table I lists the parameters used for servers, VNFs and middleboxes. In the rest of this section we use the term "middlebox" to refer to both hardware middlebox and VNF.

TABLE I
Server and Middlebox Data Used in Evaluation

Server Data [1]		
Physical CPU Cores	Idle Energy	Peak Energy
16	80.5W	2735W
Hardware Middlebox Data		
Idle Energy	Peak Energy	Processing Capacity
1100W	1700W	40Gbps
VNF Data [6], [22]		
Network Function	CPU Required	Processing Capacity
Firewall	4	900Mbps
Proxy	4	900Mbps
Nat	2	900Mbps
IDS	8	600Mbps

B. Evaluation Metrics

1) Operational Expenditure (OPEX): We measure OPEX according to Eq. 15, and compare CPLEX and heuristic by plotting the ratio of OPEX and its components.

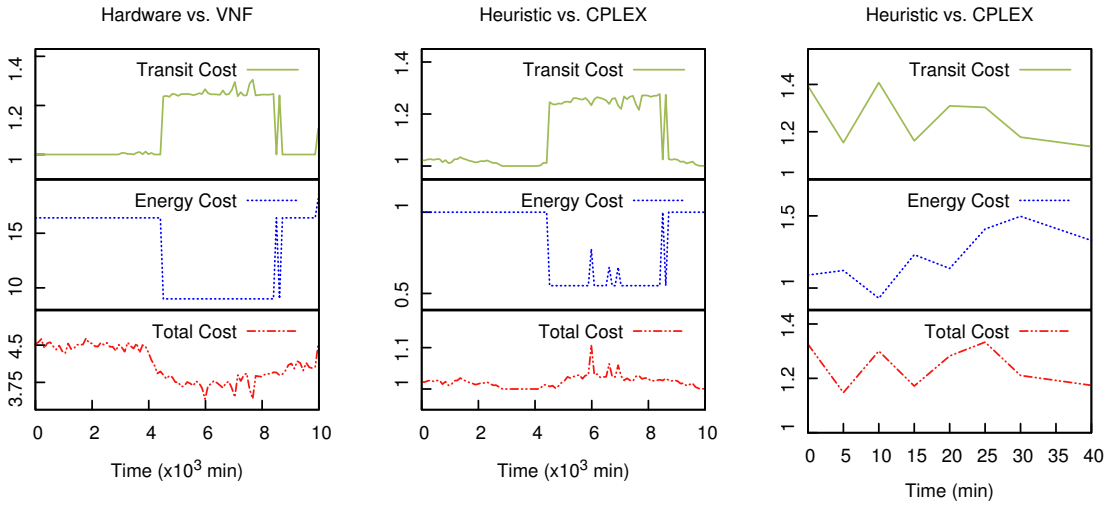
2) Execution Time: It is the time required to find middlebox placement for a given traffic batch and network topology.

3) System Utilization: We compute it as the fraction of used CPU for a server. We also report the number of active servers.

4) Topological Properties of Solution: We report two topological properties of the middlebox locations: (i) percentage of middleboxes placed withing k-hops from the ingress/egress switches and (ii) path stretch, *i.e.*, the ratio of path length obtained by CPLEX or the heuristic to the shortest path length for the traffic. The first metric gives us an insight into the location of middleboxes with respect to the ingress/egress switches, and the second one shows how many additional links (hence more bandwidth) are required to steer traffic through middlebox sequences.

C. VNFs vs. Hardware Middleboxes

One of the driving forces behind NFV is that VNFs can significantly reduce a network's OPEX. Here, we provide quantifiable results to validate this claim. Fig. 3(a) shows the ratio of OPEX for hardware middleboxes to VNFs for incoming traffic provisioning requests (about 132 requests per batch) over a period of 10000 minutes. We show two components of OPEX: energy and transit cost. There is no publicly available data that can be used to estimate the deployment cost of hardware middleboxes. So, for this experiment, we do not consider deployment cost as a component of OPEX to make the comparison fair. The SLO violation penalty is not shown as it is zero for all time-instances. We implemented a different CPLEX program to peak provision the hardware



(a) Hardware vs. VNF (Internet2) (b) Internet2 (c) Data Center

Fig. 3. Time vs. Cost Ratio

middleboxes (peak traffic occurs at time-instance 7665). VNFs are provisioned at each time-instance by the CPLEX program corresponding to the formulation provided in Section III.

The bottom part of Fig. 3(a) shows that VNFs provide more than $4\times$ reduction in OPEX. The individual reductions in energy and transit costs are also shown in the same figure. The reduction in energy cost is much higher than that of the transit cost. This is due to the fact that hardware middleboxes consume considerably higher energy than commodity servers. From Fig. 3(a) and Fig. 11(a), we can also see that with the increase in traffic volume (after time-instance 4000) the total cost ratio decreases. Interestingly, the energy cost ratio decreases, but the transit cost ratio increases. Handling higher traffic volume requires higher number of VNFs to be deployed, which increases the energy consumption of commodity servers, thus decreasing the energy cost ratio. However, VNFs are provisioned at optimal locations by CPLEX, which causes the transit cost to decrease and increases the transit cost ratio. The cost ratio relationship between VNFs and hardware middleboxes depends on a number of factors like processing capacity, traffic volume, idle and peak energy consumption.

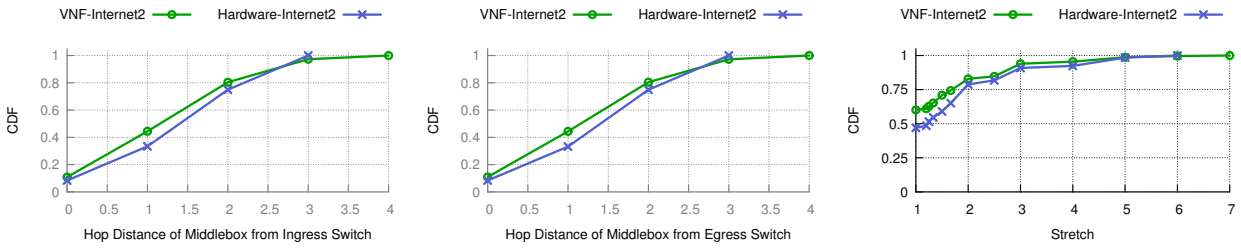
The topological properties of VNF and hardware middlebox placement locations are reported in Fig. 4. The CDF of hop distance between the ingress switch and middlebox is shown in Fig. 4(a). Higher percentage of VNFs are located within 2 hops of the ingress switch (mostly withing 1 hop), compared to hardware middleboxes. Some VNFs are also located at 4 hop distance. This only occurs when placing a VNF farther away reduces the OPEX by decreasing the energy cost. Similar results are obtained for the hop distance between middlebox and egress switch (Fig. 4(b)). These two figures also demonstrate the fact that CPLEX places middleboxes in a more balanced (symmetric) way on the path between the ingress and egress switch. The path stretch for both hardware middleboxes and VNFs are shown in Fig. 4(c). VNFs consistently achieve a lower path stretch than hardware middleboxes, as VNF locations are not static like the hardware middleboxes. They can be provisioned on any server to reduce OPEX.

D. Performance Comparison Between CPLEX and Heuristic

Now, we compare the performance of our heuristic with that of the optimal solution. Fig. 3(b) and Fig. 3(c) show the cost ratios for Internet2 and data center networks, respectively. The traffic patterns for these two topologies are shown in Fig. 11(a) and Fig. 11(b), respectively. The deployment cost and penalty for SLO violation are not shown, as the deployment cost is equal in both cases and the SLO violation penalty is zero for all time-instances. From Fig. 3(b), we can see that the heuristic finds solutions that are within 1.1 times of the optimal solution. During peak traffic periods, the ratio of energy cost goes below 1, but the ratio of transit cost increases. The optimal solution adapts to high traffic volumes by deploying more VNFs (increasing energy cost) and placing them at locations that decrease the transit cost. As a result, the ratio of energy cost decreases and the ratio of transit cost increases. However, the total cost ratio stays almost the same (varying between 1 and 1.1). Similar results are obtained for the data center network (Fig. 3(c)) as well. Here, the cost ratio is also very close to 1 and varies between 1.1 and 1.3.

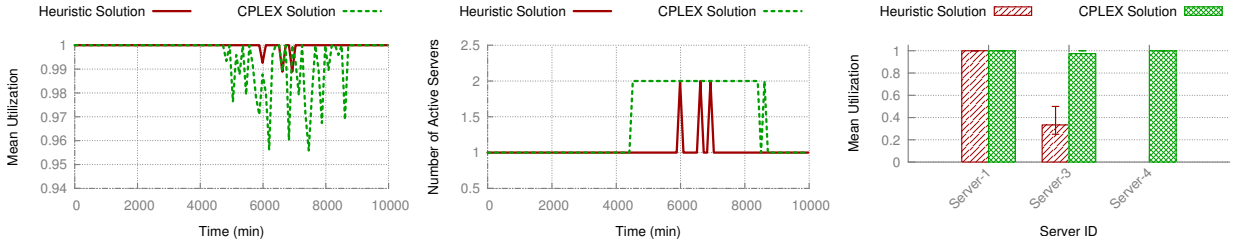
The average execution times of the heuristic and CPLEX are shown in Table II. They were run on a machine with 10×16-Core 2.40GHz Intel Xeon E7-8870 CPUs and 1TB memory. As we can see, our heuristic provides solutions that are very close to the optimal one and its execution time is several order of magnitude faster than CPLEX.

Fig. 5 shows results related to server resource utilization for Internet2 and data center networks. Fig. 5(a) and Fig. 5(b) show the mean utilization and the total number of active servers, respectively, for the Internet2 topology. Fig. 5(c) shows the average utilization per server over all time-instances.

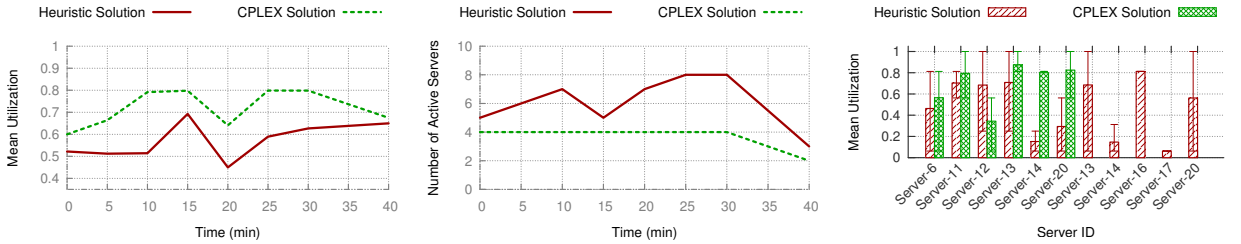


(a) Distance between Middlebox and Ingress (b) Distance between Middlebox and Egress (c) Path Stretch

Fig. 4. Topological Property Comparison between Hardware middlebox and VNF deployment (Internet2)



(a) Mean Server Utilization (Internet2) (b) Number of Active Servers (Internet2) (c) Per Server Utilization (Internet2)



(d) Mean Server Utilization (Data Center) (e) Number of Active Servers (Data Center) (f) Per Server Utilization (Data Center)

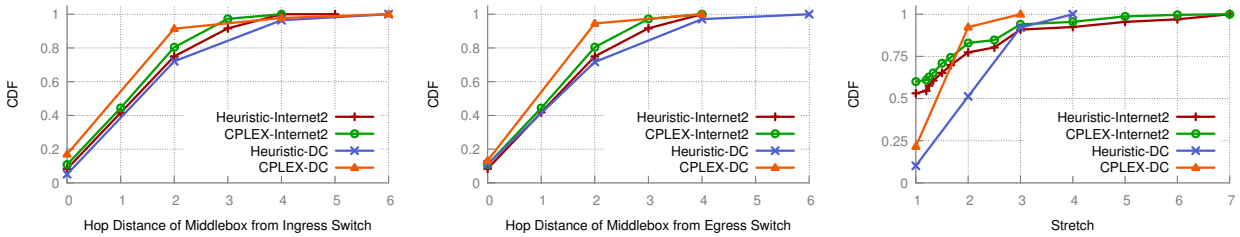
Fig. 5. Resource Utilization

TABLE II
AVERAGE EXECUTION TIME

Topology	CPLEX	Heuristic
Internet2 (12 nodes, 15 links)	34.99s	0.535s
Data Center (23 nodes, 43 links)	1595.12s	0.442s
AS-3967 (79 nodes, 147 links)	∞	2.54s

The mean utilization of the heuristic is less than that of CPLEX, as CPLEX uses more servers than the heuristic (Fig. 5(b)). CPLEX achieves lower OPEX by deploying more VNFs during higher traffic periods to route traffic through shorter paths. However, the solutions provided by the heuristic are within 1.1 times the optimal one (Fig. 3(b)). In case of the data center network, CPLEX uses less servers than the heuristic (Fig. 5(e)) and the utilization is also higher (Fig. 5(d)). The solution provided by the heuristic has higher resource fragmentation than the CPLEX one (Fig. 5(f)). The data center topology offers higher number of locations to deploy VNFs compared to Internet2. Hence, the heuristic falls a little short of the optimal placement as it explores a smaller solution space. CPLEX finds the optimal value, but at the cost of much higher execution time (Table II).

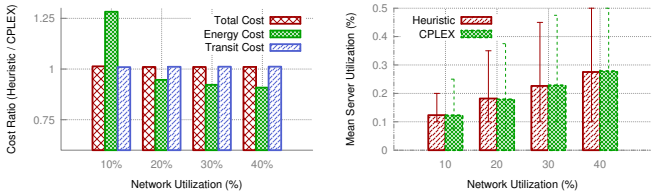
The topological properties for middlebox deployment for Internet2 and data center networks are shown in Fig. 6. The CDF of hop distance from the ingress switch to a VNF is shown in Fig. 6(a). The hop distances for the heuristic is quite close to that of the optimal solution. In case of the data center network, there is a relatively larger gap. This occurs due to the higher path diversity offered by a data center network. Each pair of nodes has more than one equal cost path. CPLEX finds the optimal solution by exploring all of them. The heuristic always picks the first shortest path. It does not explore the alternate paths to keep the execution time within practical limits (Table II). Similar results are observed for the egress case (Fig. 6(b)). From Fig. 6(a) and Fig. 6(b) we can also see that the CDFs are quite similar, which means that both CPLEX and heuristic place VNFs uniformly on the path between the ingress and egress switches. The path stretch is shown in Fig. 6(c). As before, the heuristic's performance is close to that of the optimal solution. In case of the data center network, the heuristic has a larger stretch, which is a result of the path diversity issue discussed earlier.



(a) Distance between middlebox and ingress (b) Distance between middlebox and egress (c) Path stretch

Fig. 6. Topological properties of solution

We obtained similar results for the Rocketfuel topology. Due to space limitations they are discussed in the Appendix.



(a) Cost Ratio (b) Server Utilization

Fig. 7. Cost Ratio (Heuristic / CPLEX) with Varying Load

E. Effect of High Traffic Volume

Now, we show the impact of higher traffic volume on our solutions. We perform this experiment by increasing the original traffic by 10% to 40% (in increments of 10%) for the Internet2 topology (Fig. 7). We observed a linear relationship between OPEX and network utilization for both of our solutions. The cost also grows almost at the same rate for both CPLEX and heuristic as evident from Fig. 7(a). The heuristic is able to follow the optimal solution very closely. Although it might seem a bit unintuitive by looking at the ratio of the individual cost components, it occurs as the transit cost is two order-of-magnitude larger than the energy cost.

The server utilization increases sub-linearly with increasing network load (Fig. 7(b)). The number of used servers remains the same for different network loads, but more cores were used since more VNFs were deployed. The larger error bar for CPLEX indicates the deployment of more VNFs, which increases the energy cost. However, more VNFs eventually decreased the transit cost, which is the major contributor to OPEX in this case.

VI. Related Works

The initial drive for NFV was from several telecommunication operators back in 2012 [11]. The motivation behind NFV is to break the barrier of proprietary hardwares and have more flexibility in the network in terms of the placement of service points, introducing new services, vendor independence *etc*. To this date, research efforts have been made in different aspects of NFV. In this section, we discuss about state of the art NFV management and orchestration proposals (Section VI-A) followed by some enabling technologies for NFV (Section VI-B).

A. Management and Orchestration of Network Functions

Some of the early works on managing Network Functions (NFs) in networks, propose to outsource them to a cloud service [16], [32]. Such outsourcing is motivated in the literature by studying experiences of different network operators. [16], [32] show how the management complexities arising in today's enterprise networks can be mitigated by outsourcing.

A more management approach towards NFV is taken by projects like Stratos [14], OpenNF [15]. Stratos proposes an architecture for orchestrating VNFs outsourced to a remote cloud by taking care of traffic engineering, horizontal scaling of VNFs *etc.* On the other hand, OpenNF proposes a converged control plane for VNFs and network forwarding plane by extending the centralized SDN paradigm. Some recent research works [23], [24] provide initial study on placing VNFs. However, none of the aforementioned research works address the issue of dynamically adjusting the placement of VNFs to balance between network operating cost and performance.

Some recent works on managing NFs focus on traffic engineering issues such as steering the traffic through some predefined sequence of NFs. This problem becomes more challenging when NFs along the sequence modify the packet headers, thus changing the traffic signature. Proposals like [12] and [27] propose SDN based solutions to the traffic steering problem. They propose tagging based mechanisms to identify a traffic during its lifetime and also to keep track of the visited sequence of middleboxes. The global network view of SDN makes it easier to manage and assign tags to different traffics and to ensure different policy enforcement on NFs.

B. Enabling Technologies for NFV

NFV proposes to run VNFs on commodity hardware as virtual appliances. This flexibility raises the question of performance. In recent years, a number of research efforts have been targeted to achieve near line speed network I/O throughput with commodity servers [3], [29]. Apart from accelerating the packets along the network I/O stack, more recent works have proposed changes to virtualization technologies to support the deployment of modular software NFs on lightweight VMs [22]. Hundreds of these VMs can be instantiated on a

single physical machine within miliseconds to run different VNFs. Substantial research efforts are also being put towards programming models and deployment architecture for VNFs as well. CoMb [31] and xOMB [7] propose an extensible and consolidated framework for incrementally developing scalable middleboxes. Both of these works leverage the idea of reusable network processing pipelines for middlebox composition.

VII. Conclusion

Virtualized network functions provide a flexible way to deploy, operate and orchestrate network services with much less capital and operational expenses. Software middleboxes (*e.g.*, ClickOS) are rapidly catching up with hardware middlebox performance. Network operators are already opting for NFV based solutions. We believe that our model for dynamic VNF orchestration will have significant impact on middlebox management in the near future. Our model can be used to determine the optimal number of VNFs and to place them at the optimal locations to optimize network operational cost and resource utilization. Our trace driven simulations on the Internet2 research network demonstrate that network OPEX can be reduced by a factor of 4 over hardware middleboxes through proper VNF orchestration.

In this paper, we presented two solutions to the VNF orchestration problem: CPLEX based optimal solution for small networks and a heuristic for larger networks. We found that the heuristic produces solutions that are within 1.3 times of the optimal solution, yet the execution-time is about 65 to 3500 times faster than that of the CPLEX solution. We intend to extend this work in a number of ways. We plan to extend our model for supporting both hardware and software middleboxes in the same network. We want to explore the possibility of introducing failure-resilience by deploying backup VNFs that can take over the traffic processing tasks from failed VNFs. We also plan to enhance the physical network transformation process to further reduce the solution space and speed-up the running time of the optimal solution.

References

[1] Comparison of enterprise class power enclosure. http://www.dell.com/downloads/global/products/pedge/en/bladepower-studywhitepaper_08112010_final.pdf.

[2] https://datatracker.ietf.org/doc/draft-ietf-sfc-dc-use-cases/.

[3] Intel DPDK. http://dpdk.org/.

[4] Internet Transit Pricing. http://drpeering.net/white-papers/Internet-Transit-Pricing-Historical-And-Projected.php.

[5] Internet2 Research Network Topology and Traffic Matrix. http://www.cs.utexas.edu/~yzhang/research/AbileneTM/.

[6] pfSense Hardware Sizing Guide. https://www.pfsense.org/hardware/#sizing.

[7] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat. xOMB: Extensible open middleboxes with commodity servers. In *Proc. of ACM/IEEE ANCS '12*, pages 49–60.

[8] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of ACM IMC '10*, pages 267–280.

[9] C.-C. Chiou. Transshipment problems in supply chain systems: review and extensions. *Supply Chain*, pages 427–448, 2008.

[10] N. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862 – 876, 2010.

[11] ETSI. Network Functions Virtualisation – Introductory White Paper. https://portal.etsi.org/NFV/NFV_White_Paper.pdf, 2012.

[12] 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. of USENIX NSDI '14*.

[13] G. D. Forney Jr. The Viterbi Algorithm. *Proc. of the IEEE*, 61(3):268–278, 1973.

[14] A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, and A. Akella. Stratos: A Network-Aware Orchestration Layer for Virtual Middleboxes in Clouds. *arXiv preprint arXiv:1305.0209*, 2013.

[15] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. OpenNF: enabling innovation in network function control. In *Proc. of ACM SIGCOMM '14*, pages 163–174.

[16] G. Gibb, H. Zeng, and N. McKeown. Outsourcing network functionality. In *Proc. of ACM HotSDN '12*, pages 73–78.

[17] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro. Service Function Chaining Use Cases in Mobile Networks, 2014.

[18] J. Hwang, K. K. Ramakrishnan, and T. Wood. NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms. In *Proc. of USENIX NSDI '14*, pages 445–458.

[19] L. T. Kou and G. Markowsky. Multidimensional bin packing algorithms. *IBM Journal of Research and development*, 21(5):443–448, 1977.

[20] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *arXiv preprint arXiv:1406.0440*, 2014.

[21] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Q. Fu, Q. Sun, C. Pham, C. Huang, J. Zhu, and P. He. Service Function Chaining Problem Statement. *draft-liu-sfc-use-cases-08 (work in progress)*, 2014.

[22] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. ClickOS and the art of network function virtualization. In *Proc. of USENIX NSDI '14*, pages 459–473.

[23] S. Mehraghdam, M. Keller, and H. Karl. Specifying and Placing Chains of Virtual Network Functions. *arXiv preprint arXiv:1406.1058*, 2014.

[24] H. Moens and F. De Turck. VNF-P: A Model for Efficient Placement of Virtualized Network Functions. In *Proc. of ManSDN/NFV '14*.

[25] A. Nucci, A. Sridharan, and N. Taft. The problem of synthetically generating ip traffic matrices: initial recommendations. *ACM CCR*, 35(3):19–32, 2005.

[26] H. Pirkul and V. Jayaraman. A Multi-commodity, Multi-plant, Capacitated Facility Location Problem: Formulation and Efficient Heuristic Solution. *Comput. Oper. Res.*, 25(10):869–878, Oct. 1998.

[27] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *Proc. of ACM SIGCOMM '13*, pages 27–38.

[28] P. Quinn and T. Nadeau. Service Function Chaining Problem Statement. *draft-quinn-sfc-problem-statement-10 (work in progress)*, 2014.

[29] L. Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *Proc. of USENIX ATC '12*, pages 101–112.

[30] L. Saino, C. Cocora, and G. Pavlou. A Toolchain for Simplifying Network Simulation Setup. In *Proc. of SIMUTOOLS '13*.

[31] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and Implementation of a Consolidated Middlebox Architecture. In *Proc. of USENIX NSDI '12*, pages 323–336.

[32] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM CCR*, 42(4):13–24, 2012.

[33] J. Sherry and S. Ratnasamy. A Survey of Enterprise Middlebox Deployments. Technical Report UCB/EECS-2012-24, EECS Department, University of California, Berkeley, Feb 2012.

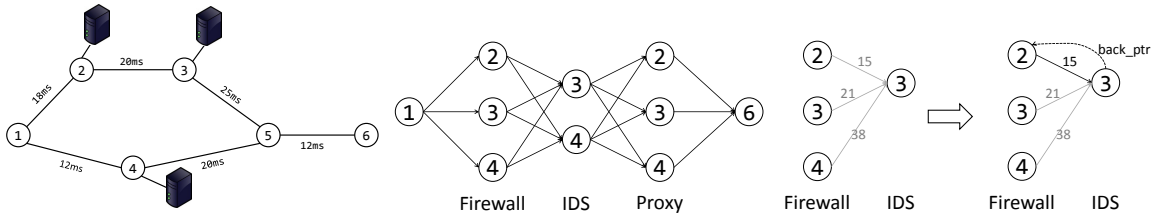
[34] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. 32(4):133–145, 2002.

[35] S. Surendra, M. Tufail, S. Majee, C. Captari, and S. Homma. Service Function Chaining Use Cases in Mobile Networks, 2014.

Appendix A
Glossary of Symbols

Appendix B
Heuristic Algorithm

Algorithm 1 gives the pseudcode of the heuristic solution. The procedure $ProvisionTraffic$ takes as input a traffic request t and the topology graph $\bar{G}$ annotated with the resource capacities at each switch. We keep two tables, $cost$ and π,



(a) Example Network Topolog (b) Multi-Stage Graph (c) A Single Stage

Fig. 8. Modeling with Multi-Stage Graph

Physical Network	
$G(S, L)$	Physical network G with switches S and links L
N	Set of servers
$h_{\bar{n}\bar{s}} \in \{0, 1\}$	If server $\bar{n} \in N$ is attached to switch $\bar{s} \in S$
R	Set of resources offered by servers
$c^r_{\bar{n}} \in \mathbb{R}^+$	Resource capacity of server $\bar{n}$, $\forall r \in R$
$\beta_{\bar{u}\bar{v}}, \delta_{\bar{u}\bar{v}} \in \mathbb{R}^+$	Bandwidth, propagation delay of link $(\bar{u}, \bar{v}) \in L$
$\eta(\bar{u})$	Neighbors of switch $\bar{u}$
Virtualized Network Functions (VNFs)	
P	Set of possible VNF types
$\mathcal{D}^+_p, \kappa^r_p, c_p, \delta_p$	Deployment cost, resource requirement, processing capacity and processing delay of VNF type $p \in P$
$d_{\bar{n}p} \in \{0, 1\}$	If VNF type p can be provisioned on server $\bar{n}$
Traffic	
$\bar{u}^t, \bar{v}^t, \Psi^t$	Ingress, egress and VNF sequence for traffic t
$\beta^t, \delta^t, \omega^t$	Bandwidth, expected delay, SLA penalty for t
N^t	$\{\bar{u}^t, \bar{v}^t, \Psi^t\}$
L^t	$\{(\bar{u}^t, \Psi^t_1), \ldots, (\Psi^t_{\lvert\Psi^t\rvert-1}, \Psi^t_{\lvert\Psi^t\rvert}), (\Psi^t_{\lvert\Psi^t\rvert}, \bar{v}^t)\}$
$\eta^t(n)$	Neighbors of $n \in N^t$
$g^t_{np} \in \{0, 1\}$	If node $n \in N^t$ is of type $p \in P$
$\mathcal{M}$	Set of pseudo-VNFs
$\zeta(m)$	$\zeta(m) = \bar{n}$ if VNF $m \in \mathcal{M}$ is attached to server $\bar{n}$
$\Omega(\bar{n})$	$\{m \mid \zeta(m) = \bar{n}\},\ m \in \mathcal{M}, \bar{n} \in N$
$q_{mp} \in \{0, 1\}$	If VNF $m \in \mathcal{M}$ is of type $p \in P$
Decision Variables	
$^*x^t_{nm} \in \{0, 1\}$	If node $n \in N^t$ is provisioned on $m \in \mathcal{M}$
$^*w^{tn_1n_2}_{\bar{u}\bar{v}}$	If $(n1, n2) \in L^t$ uses physical link $(\bar{u}, \bar{v}) \in L$
Derived Variables	
$^*y_m \in \{0, 1\}$	If VNF $m \in \mathcal{M}$ is active
$z^t_{n\bar{s}} \in \{0, 1\}$	If node $n \in N^t$ is attached to switch $\bar{s}$
$^*\hat{x}^t_{nm}, \hat{w}^{tn_1n_2}_{\bar{u}\bar{v}}, \hat{y}_m$	denote value from the previous iteration

to keep track of the cost and the sequence of middlebox placements, respectively. $cost_{i,j}$ represents the cost of deploying the j-th middlebox in the middlebox sequence Ψ^t to a server attached with switch i. The cost computation procedure is the same as described in Section IV-B. We use a number of helper procedures for the ease of implementation. The first helper procedure, $IsResourceAvailable$ checks if a middlebox $mbox$ for a traffic request t can be placed at a switch i, satisfying the minimum bandwidth and resource requirements. The second helper, $GetCost$, computes the cost of placing middlebox $mbox$ for a traffic request t at a server attached to switch j. The previous node k that yields the minimum cost for the current node in consideration j, is tracked by the entry $\pi_{k,j}$. Finally, we backtrace using entries in π to obtain the desired middlebox sequence.

Running Time: Let the number of switches and the maximum length of a middlebox sequence be n and m,

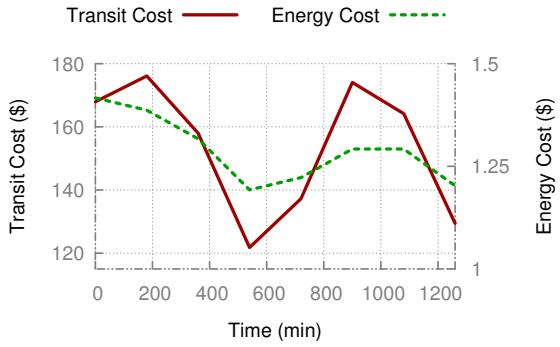


Fig. 9. OPEX Components for AS-3967

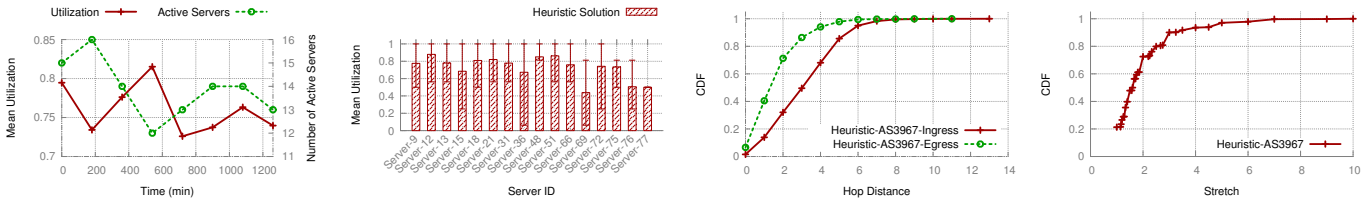
respectively. Algorithm 1 performs $\Theta(nm)$ computations at the beginning to initialize the cost matrix. Then for each element in the traffic sequence, the algorithm takes all possible pairs of nodes u, v and computes the cost of deploying a middlebox at the server attached to switch v given that the previous middlebox in the sequence was deployed at a server connected to switch u. Therefore, there is a total of $\Theta(n^2m)$ operations involved. With some pre-computation steps the costs can be calculated and resource availability can be queried in $O(1)$ time. Therefore, Algorithm 1 runs in $\Theta(n^2m)$.

Appendix C
Heuristic in Action

Fig. 8(a) shows an example network topology with six switches, where the servers are connected to switch 2, 3 and 4. Now, we are required to find the path for a traffic which is going from switch 1 to 6 and must pass through a firewall, then an IDS and finally through a proxy.

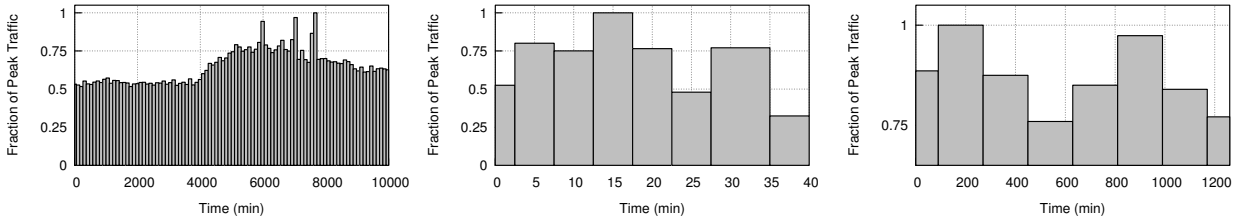
Now, we generate a multi-stage graph as shown in Fig. 8(b). Here, we are assuming that the firewall and proxy can be deployed on any server, but the IDS can only be deployed on servers connected to switch 3 and 4. Each node in the multi-stage graph represents a decision about where to place a VNF. For example, if we select node 4 in the stage labeled "IDS", it means that a VNF corresponding to an IDS will be deployed on the server connected to switch 4. As explained earlier, there is a cost associated with each node selection.

Now, we traverse this graph starting at node 1. The first stage is trivial, we just compute the cost of deploying and running (energy cost) a firewall at node 2, 3 and 4 and add



(a) Mean Server Utilization (b) Per Server Utilization (c) Distance to middlebox (d) Path stretch

Fig. 10. Results for Rocketfuel Topology (AS-3967)



(a) Internet2 (b) Data Center (c) Rocketfuel

Fig. 11. Traffic Distribution over Time for Different Scenarios

the cost of routing traffic from node 1 to each node. There is no additional computation as there is just one incoming link for each node. However, the operations for the subsequent stages involve comparing the cost of reaching a particular node from different nodes. For example, node 3 in stage "IDS" can be reached from three different nodes. The operation performed in this is stage is explained in Fig. 8(c).

We need to compute the cost of transition from nodes 2, 3 and 4 to node 3. These costs are shown on the left side of Fig. 8(c). Now, if we select the link between node 4 and node 3 then the Firewall will be deployed on node 4 and the IDS will be deployed on node 3 and cost of deploying the IDS will be 38. However, we have links with lower costs than this one and at each stage we select the incoming link with the minimal cost. So, here we will select the link between node 2 and 3 as it has the lowest cost of 15. We will also save a pointer (`back_ptr`) to mark the node that was selected. We continue in this manner until we reach the destination node (node 6 in this example), then we follow the `back_ptrs` to re-construct the solution.

Appendix D
Results for Rocketfuel Topology

The results for the AS-3967 topology is shown in Fig. 9 and Fig. 10. The traffic for this topology is show in Fig. 11(c). As mentioned earlier, this traffic was generated using the FNSS tool [30] and it exhibits time-of-day effect. We cannot provide a comparison with the optimal solution as the CPLEX program was not able to solve the problem for this topology. It failed to fit the optimization model in its memory even though the physical machine had 1TB of memory. The program crashes after the total memory usage reaches around 300 GB. We observed similar behavior when experimenting with high traffic volumes. CPLEX was not able to solve the problem for the Internet2 topology when traffic was increased to utilize the network by more than 40%. We tuned different parameters (*e.g.*, solving the dual problem, storing branch and bound tree data on disk, reducing the number of threads, *etc.*) of the CPLEX solver according to the guidelines provided by IBM[1], but could not solve the problem. We plan to investigate this issue further in the future. However, the heuristic solution was able to solve the same problem in less than 3 seconds.

The transit and energy cost for the AS-3967 topology is reported in Fig. 9. The transit cost is two order-of-magnitude higher than the energy cost, which is expected for a larger network with large amount of traffic. From Fig. 11(c) and Fig. 9, we can see that our dynamic VNF orchestration approach adapts nicely with the changing traffic conditions. It can dynamically scale-up or scale-down the number of active VNFs (demonstrated by the rise and fall of the energy cost). It can also adapt the location of the VNFs according to the variation in the traffic volume.

The results for system resource utilization and topological properties for middlebox locations are shown in Fig. 10. From Fig. 10(a) we can see that the mean utilization and number of active servers vary with fluctuation in traffic volume. The mean utilization of the servers is around 80%, but there is a small number of servers that are underutilized (Fig. 10(b)). The CDF of percentage of middleboxes deployed within k-hop distance from the ingress switch is reported in Fig. 10(c). More than 90% middleboxes are deployed within 5 hops, which is quite reasonable for a network with 79 switches and 147 links. Similar results are obtained for the egress case as shown in the same figure. Finally, the path stretch is shown in Fig. 10(d). We

[1]http://www-01.ibm.com/support/docview.wss?uid=swg21399933

can observe that 20% traffic passes through the shortest path even after going though the VNF sequence. So, in 20% of the cases VNFs are provisioned on the shortest path between the ingress and egress switches that the traffic is passing through.