On Orchestrating Virtual Network Functions

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 Virtual Network Functions (VNFs). It is a challenging problem to determine the required number and placement of VNFs that optimize 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 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 middleboxes is comparable to that of routers in an enterprise network [24]. However, middleboxes come with high Capital Expenditures (CAPEX) and Operational Expenditures (OPEX). They are expensive, vendor-specific, and require specially trained personnel for configuration and maintenance. Moreover, it is often impossible to add new functionalities to an existing middlebox, which makes it very difficult for network operators to deploy new services.

Another problem arises 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 a firewall, then an IDS, and finally through a proxy [21]. This phenomenon is typically referred to as Service Function Chaining (SFC) [22]. Several IETF drafts demonstrate middlebox chaining use-cases in operator networks [14], mobile networks [12], and data center networks [26]. Currently, this task is performed by manually crafting the routing table entries. It is a cumbersome and error-prone process. Moreover, a fixed placement cannot be optimal for all possible traffic patterns in the long run.

An emerging and promising technology that can address these limitations is Network Function Virtualization (NFV) [8]. It proposes to move packet processing from hardware middleboxes to software middleboxes or Virtual Network Functions (VNFs). Instead of running hardware middleboxes, the same packet processing tasks are performed by software running on commodity servers. This approach does not hamper performance as many state-of-the-art software middleboxes have already achieved near-hardware performance [13], [17]. NFV provides ample 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.

VNF chains can be orchestrated by dynamically deploying a composition of VNFs either on a single server or a cluster of servers. This approach can significantly reduce the OPEX of a network. However, several issues must be considered during VNF provisioning: (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. An optimal VNF orchestration strategy must address these issues during the cost minimization process. Moreover, it must avoid penalties for Service Level Objective (SLO) violations and satisfy the capacity constraints of the physical servers and physical links. We refer to this problem as the *Virtual 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 finds solutions within 1.3 times of the optimal for 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). Finally, we conclude in Section VII.

978-3-901882-77-7

II. Mathematical Model and Problem Definition

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. Virtual 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 $p \in P$ 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. Each VNF type has 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 path setup requests for different kinds of traffic. A traffic request is represented by a 5-tuple $t = \langle \bar{u}^t, \bar{v}^t, \Psi^t, \beta^t, \delta^t \rangle$, 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) and l_{Ψ^t} denotes the length of the VNF sequence Ψ^t.

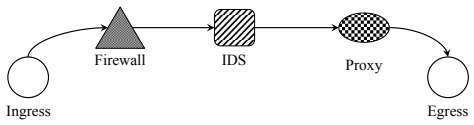


Fig. 1. Traffic Model

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 or 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)

Here, we are given a physical network topology, VNF specifications, network status, and a set of traffic requests. Our objective is to minimize the overall network OPEX by (i) provisioning optimal number of VNFs, (ii) placing them at optimal locations, and (ii) finding optimal routing paths, while respecting the capacity constraints (*e.g.*, physical servers, links, VNFs), delay constraints, and ensuring proper VNF sequence.

III. Integer Linear Programming (ILP) Formulation

VNF-OP is a considerably harder problem to solve than traditional Virtual Network (VN) embedding problems [7], as virtual resources are shared between multiple requests. In this work, we address these challenges by judiciously augmenting the physical network as 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 VNF type that can be deployed on each server. We calculate this number based on the resource capacities of the server and the resource requirements of a VNF type. 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 the pseudo-VNFs attached to server $\bar{n}$:

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

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

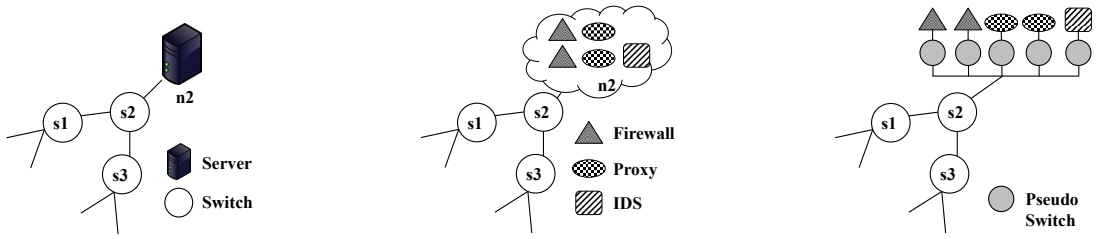
$$q_{mp} = \begin{cases} 1 & \text{if VNF } m \text{ is of type } p \in P, \\ 0 & \text{otherwise.} \end{cases}$$

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.

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



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

Fig. 2. Network Transformation

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 attached. 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:

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

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

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

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

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

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

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

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

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

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

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

Now, we present the capacity constraint for physical links:

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

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:

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

To meet SLOs, the total propagation delay experienced by traffic t must be within the expected propagation delay δ^t:

$$\sum_{n_1\in N^t}\ \sum_{\substack{n_2\in\eta^t(n_1) \\ \text{and } n_2>n_1}}\ \sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\eta(\bar{u})} w^{tn_1n_2}_{\bar{u}\bar{v}}\delta_{\bar{u}\bar{v}} \leq \delta^t \quad (10)$$

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

$$\sum_{\bar{u}\in\bar{S}}\sum_{\bar{v}\in\bar{S}} (w^{tn_1n_2}_{\bar{u}\bar{v}} + w^{tn_1n_2}_{\bar{v}\bar{u}}) \geq 0,$$
$$\forall\ t \in T, n_1, n_2 \in N^t | n_2 \in \eta^t(n_1) \text{ and } n_2 > n_1 \quad (11)$$

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

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

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

$$\mathbb{D} = \sum_{m \in \mathcal{M} | y_m = 1} \mathcal{D}_p^+ \times q_{mp} \times (y_m - \hat{y}_m) \tag{12}$$

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 r, respectively.

$$\mathbb{E} = \sum_{\bar{n} \in \bar{N}} \sum_{m \in \Omega \bar{n}} y_m \times q_{mp} \times e^r(c^r_{\bar{n}}, \kappa^r_p) \tag{13}$$

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). So, cost of traffic forwarding:

$$\mathbb{F} = \sum_{\substack{t \in T \\ n_1 \in N^t}} \sum_{\substack{n_2 \in \eta^t(n_1) \\ \text{and } n_2 > n_1}} \sum_{\substack{\bar{u} \in \bar{S} \\ \bar{v} \in \eta(\bar{u})}} (w^{tn_1n_2}_{\bar{u}\bar{v}} - \hat{w}^{tn_1n_2}_{\bar{u}\bar{v}}) \times \beta^t \times \sigma \tag{14}$$

Our objective is to minimize the weighted sum of the aforementioned costs.

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

Here, α, β, and γ are weighting factors that are used to adjust the relative importance of the cost components. We can reduce the *Multi-Commodity, Multi-Plant, Capacitated Facility Location Problem (MCMP-CFLP)* [6], [20]) to VNF-OP, which is known to be NP-Hard. Hence, VNF-OP is also NP-Hard. In the next section, we propose a heuristic algorithm.

IV. Heuristic Solution

Given a network topology, a set of middlebox specifications, and a batch of traffic requests, the heuristic finds the number and locations of different VNF types required to operate the network with minimal OPEX. The heuristic runs in two steps: (i) we model the VNF-OP as a multi-stage directed graph with associated costs, (ii) we find a near-optimal solution from the multi-stage graph by running the Viterbi algorithm [9].

A. Modeling with Multi-Stage Graph

For a given traffic request, $t = \langle \bar{u}^t, \bar{v}^t, \Psi^t, \beta^t, \delta^t \rangle$, 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 ($\forall i \in \{2, \ldots (l_{\Psi^t}+1)\}$), 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. 12) and an energy cost (Eq. 13) as described in Section III-B.

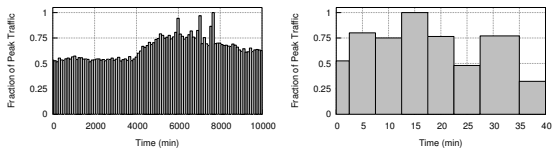
An edge $(\bar{v}_i, \bar{v}_j)$ in this multi-stage graph represents the placement of a VNF at a server attached to switch $\bar{v}_j$, given that the previous VNF in the sequence is deployed on a server attached to switch $\bar{v}_i$. We put a directed edge between all pairs of nodes in stage i and $i+1$ ($\forall i \in \{1, 2, \ldots (l_{\Psi^t} + 1)\}$). We associate the cost for forwarding traffic (Eq. 14) with each edge. The traffic forwarding cost is proportional to the weighted shortest path (in terms of latency) between the switches. 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 and then computes 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. This computation proceeds in increasing order of the stages. 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 $cost_{v,u}$ (defined according to Section IV-A). We keep track of the minimum cost path using the table π. The desired VNF placement is constructed by back tracing pointers from the final stage to the first stage using π. For each traffic request, the heuristic runs in $\Theta(n^2m)$

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

$\forall (i,j) \in \{1 \ldots |\Psi^t|\} \times \{1 \ldots |\bar{S}|\} : cost_{i,j} \leftarrow \infty, \pi_{i,j} \leftarrow NIL$
$\forall i \in |\bar{S}|$:
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
$\forall (i,j,k) \in \{2 \ldots |\Psi^t|\} \times \{1 \ldots |\bar{S}| \times \{1 \ldots |\bar{S}|\}$:
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 $cost_{|\Psi^t|,i}$
$\forall i \in < |\Psi^t|, |\Psi^t| - 1 \ldots 1 >$: Append Π to ψ, $\Pi \leftarrow \pi_{i,\Pi}$
return $Reverse(\psi)$



(a) Internet2 (b) Data Center

Fig. 4. Traffic Distribution over Time for Different Scenarios

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 [4], [17]		
Network Function	CPU Required	Processing Capacity
Firewall	4	900Mbps
Proxy	4	900Mbps
Nat	2	900Mbps
IDS	8	600Mbps

time, where n is the number of switches in the network and m is the VNF sequence length.

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 optimal solution (Section V-D), and (iii) comparing the performance of our heuristic with state-of-the art (Section V-E). Before presenting the results, we briefly describe the simulation setup (Section V-A) and the evaluation metrics (Section V-B). Source code is available at `http://goo.gl/Da7EZu`.

A. Simulation Setup

1) Topology Dataset: We used two types of networks: (i) Internet2 research network (12 nodes, 15 links) [3], and (ii) A university data center network (23 nodes, 42 links) [5].

2) Traffic Dataset: We use real traffic traces for the evaluation. We use traffic matrix traces from [3] 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 [5], and replay the traffic between random source-destination pairs.

3) Middlebox and Cost Data: We have generated a 3-length middlebox sequence for each traffic based on the data provided in [2], [21]. We have used publicly available data sheets from manufacturers and service providers to select and infer values for server energy cost and 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.

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.

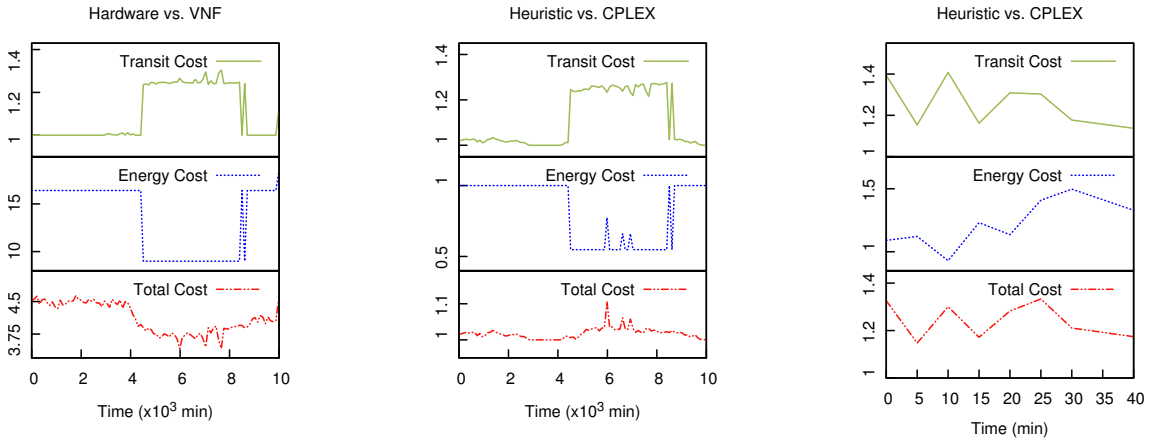
C. VNFs vs. Hardware Middleboxes

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. We implemented a separate CPLEX program that finds the optimal number and location of hardware middleboxes for the same traffic. This program finds the optimal values over all time-instances. 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. 4(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, which causes the transit cost to decrease and increases the transit cost ratio.

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. 4(a) and Fig. 4(b), respectively. The deployment cost is not shown as it is equal in both cases. 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



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

Fig. 3. Time vs. Cost Ratio

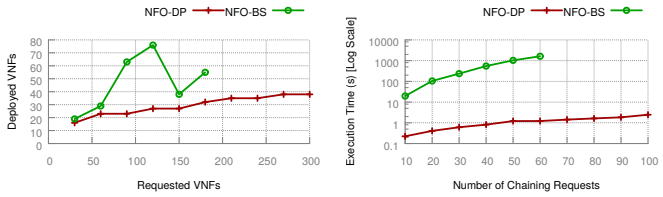
TABLE II
AVERAGE EXECUTION TIME

Topology	CPLEX	Heuristic
Internet2 (12 nodes, 15 links)	34.99s	0.535s
Data Center (23 nodes, 43 links)	1595.12s	0.442s

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.

E. Performance Comparison with Previous Work



(a) VNFs Deployed (b) Execution Time

Fig. 5. Performance comparison with [16]

We demonstrate the effectiveness of our proposed heuristic (NFO-DP) over prior work by comparing with a very recent and relevant one. We implemented the binary search based heuristic proposed in [16] (NFO-BS) and ran it on a machine with similar configurations. We adjusted the heuristic parameters according to the provided guideline in the paper. We experimented with a moderate sized ISP network topology with 79 nodes and 147 links (AS3967 from RocketFuel ISP topologies [25]). We varied the number of VNF chaining requests from 10 to 100 and measured the execution time along with the number of deployed VNFs. The results are reported in Fig. 5. The binary search based approach could not find a feasible solutions within the set time limits for more than 60 chains. Our findings show that, on a similar problem instance our proposed heuristic outperforms the state-of-the art in both solution quality and execution time.

VI. RELATED WORKS

The initial drive for NFV came from several telecommunication operators back in 2012 [8]. More recently, Management solutions for NFV are proposed by projects like Stratos [10], OpenNF [11]. Stratos proposes an architecture for orchestrating VNFs to a remote cloud with focus on traffic engineering and horizontal scaling of VNFs. OpenNF proposes a converged control plane for VNFs and network forwarding plane by extending the centralized SDN paradigm. An OpenStack based VNF orchestration system is presented in [15]. It proposes to modify the compute and networking engine of OpenStack to support intelligent placement of VNFs in VMs and transparently deploy service chains. Initial studies on placement of VNFs and VNF chains in both IP and optical networks is presented in [16], [18], [19], [23], [27]. 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.

VII. CONCLUSION

Virtualized network functions provide a flexible way to deploy, operate and orchestrate network services with much less capital and operational expenses. 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.

VIII. ACKNOWLEDGMENTS

This work was supported by the Natural Science and Engineering Council of Canada (NSERC) under the Smart Applications on Virtual Infrastructure (SAVI) Research Network.

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

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

[5] 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.

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

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] F. Lucrezia, G. Marchetto, F. G. O. Risso, and V. Vercellone. Introducing network-aware scheduling capabilities in openstack. In *Proc. of NetSoft '15*. IEEE, 2015.

[16] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions. In *IFIP/IEEE IM*. IFIP/IEEE, 2015.

[17] 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.

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

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

[20] 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.

[21] 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.

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

[23] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet. Network service chaining with efficient network function mapping based on service decompositions. In *Proc. of NetSoft '15*. IEEE, 2015.

[24] 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.

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

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

[27] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs. Network function placement for nfv chaining in packet/optical datacenters. *Lightwave Technology, Journal of*, 33(8):1565–1570, April 2015.