2017 IEEE 37th International Conference on Distributed Computing Systems

A Scalable and Distributed Approach for NFV Service Chain Cost Minimization

Zijun Zhang[§†], Zongpeng Li[§†], Chuan Wu[‡], Chuanhe Huang[§]
[§] State Key Lab of Software Engineering, School of Computer, Wuhan University
[†] Department of Computer Science, University of Calgary, {*zijun.zhang, zongpeng*}*@ucalgary.ca*
[‡] Department of Computer Science, The University of Hong Kong, *cwu@cs.hku.hk*

***Abstract*—Network function virtualization (NFV) represents the latest technology advancement in network service provisioning. Traditional hardware middleboxes are replaced by software programs running on industry standard servers and virtual machines, for service agility, flexibility, and cost reduction. NFV users are provisioned with service chains composed of virtual network functions (VNFs). A fundamental problem in NFV service chain provisioning is to satisfy user demands with minimum system-wide cost. We jointly consider two types of cost in this work: nodal resource cost and link delay cost, and formulate the service chain provisioning problem using nonlinear optimization. Through the method of auxiliary variables, we transform the optimization problem into its separable form, and then apply the alternating direction method of multipliers (ADMM) to design scalable and fully distributed solutions. Through simulation studies, we verify the convergence and efficacy of our distributed algorithm design.**

I. Introduction

Traditionally, building a service chain for a new network application requires purchasing and configuring specialized hardware devices, and physically cabling them into a specific sequence. The cost of building and maintaining such a system can be high. Furthermore, such a hardware solution is often over-provisioned, to meet the highest possible application load that occurs rather rarely in practice [1]. However, over-provisioning leads to waste of hardware resources in non-peak periods.

Network function virtualization (NFV), a paradigm shift promoted by industry players exemplified by AT&T, China Mobile and Vodafone, aims to address the above challenges by simplifying and accelerating the deployment of network services [2]. Since 2012, the European Telecommunications Standards Institute (ETSI) has published a series of white papers on NFV, covering the opportunities and challenges [2], use cases [3], architectural framework [4], and industry progresses [5]. The virtual network functions (VNFs) are instantiated on demand without the installation of new equipments [6], which enables network operators to flexibly create, upgrade and destroy service chains in a flexible and inexpensive way.

Apart from dynamic provisioning of elastic services, NFV transforms the deployment of service chains from a centralized fashion to a distributed fashion [7, 8], *i.e.*, VNFs can be instantiated on geo-distributed network points of presence (N-PoP) [9] connected by the network infrastructure. Examples of N-PoP locations include central offices, customer premises, mobile devices, and data centres. Distributed NFV enables service providers to take full advantage of existing hardware resources in different locations, enhancing service availability and reliability.

In this work, we address one of the main use cases of NFV, *i.e.*, virtual network function as a service (VNFaaS) (*e.g.*, in white paper [3]). To cater for complex demand from consumers in practice, multiple types of VNFs are combined to form a service chain, through which customer packets traverse. Since different VNF instances can run on multiple server nodes at geo-disperse locations, a fundamental question in NFV service provisioning is where to instantiate the VNFs of a service chain, in order to achieve the minimum cost of resources, while keeping the overall latency of the service chain low.

To tackle such a service chain cost minimization problem, we first construct a general optimization model for service chain delivery, VNF placement, and resource allocation, which captures diversity in location, resource cost, and latency tolerance. Second, we formulate the cost minimization problem into a convex optimization problem with linear constraints, together with a linear or nonlinear objective function. As we will see in Sec. III, the number of decision variables scales quadratically with the number of server nodes, and a centralized sequential solution to this problem does not scale well. Moreover, the cost and the available amount of resources can be private information of each server node, while a centralized approach requires gathering all such information. A centralized approach is further prone to connection failures. In comparison, a distributed approach naturally provides greater scalability and reliability.

Towards scalability, confidentiality and robustness in algorithm design, we first aim at splitting the service chain cost minimization problem into multiple subproblems, each of which corresponds to a server node or a source node, and can be solved in parallel. However, the formulated optimization problem is not separable as is. We reformulate a separable version using the method of auxiliary variables, and then develop a distributed algorithm for this problem based on the alternating direction method of multipliers (ADMM). ADMM is a simple yet powerful type of optimization algorithm dated back to the 1970s [10]. It witnessed a recent renaissance in the context of distributed convex optimization over big data, and in particular in large-scale problems arising from statistics,

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.210

machine learning [11], and cloud computing [12].

The rest of the paper is organized as follows. We review related literature in Sec. II, and define our NFV system model in Sec. III. Sec. IV presents the ADMM based distributed algorithm. Sec. V contains simulation studies. Finally, Sec. VI concludes the paper.

II. Related Work

A growing community is working intensively on developing standards for NFV, as well as the required implementation and management techniques. Han et al. [6] explain the requirements and architectural framework of NFV, present several use cases and discuss the challenges and future directions. Hawilo et al. [13] discuss the use of NFV in mobile networks. John et al. [8] summarize research directions in network service chaining, including service chain description, programming, deployment, debugging and security.

Moens and De Turck [14] focus on VNF placement in a hybrid scenario, where some of the services are provided through dedicated physical hardware, and some through virtualized service instances. They do not address the delay or resource cost. A similar VNF placement problem is discussed in [7], which aims to minimize the distance cost between customers and the VNF instances by which they are served, as well as the setup cost of these instances. However, service chaining, as an important application scenario, is not modelled in [7]. The above studies do not take into account the diversity in resource cost, or latency tolerance. Furthermore, to the authors' knowledge, no distributed algorithm for service chain provisioning or VNF placement has been proposed in the NFV literature.

ADMM made its debut as a simple yet powerful optimization method a few decades ago [10], and was recently revisited by Boyd et al. [11]. It has been widely used in statistics and machine learning, such as in [15] and [16]. ADMM was first applied to the field of cloud computing by [12], which considers the problem of joint request mapping and response routing in geo-distributed datacenters. In this work, to fit our NFV optimization problem to the standard form of ADMM, and to decompose the problem for parallel solutions, we relax the coupling caused by data flows by introducing a set of auxiliary variables. Our technique is effective on a variety of convex optimization problems that involve data flow routing.

III. System Models

A. Service Chain Modelling

We consider a NFV service provider who owns multiple types of resources (CPU, RAM, disk storage and network bandwidth) denoted by $\mathcal{R}$, on a set of server nodes $\mathcal{I}$ distributed in distinct geographical locations. Let C_i^r be the capacity of resource r on server node i. The server nodes are connected to the network infrastructure, and each pair of them can communicate with inbound (outbound) bandwidth capacity $C_i^{r_{in}}$ ($C_i^{r_{out}}$), where $i \in \mathcal{I}$, $r_{in}, r_{out} \in \mathcal{R}$. The server nodes are assumed to be fully connected in order to cover the most general case. However, there are some circumstances where the server nodes are only locally connected, which we can model by simply clamping the flows between any pair of disconnected server nodes to zero.

VNF users in a set $\mathcal{U}$ each demand a customized service chain of connected VNFs. The set of available types of VNFs is $\mathcal{N}$. For simplicity, let $u \in \mathcal{U}$ also denote the source of the corresponding service chain of user u, and $u' \in \mathcal{U}'$ the sink of the chain. Each source injects a data flow at rate f^u into its service chain. Let $\mathcal{L}(u) = \{(u, n_1), (n_1, n_2), \ldots, (n_{k-1}, n_k), (n_k, u')\}$ be the links of service chain u, where $\mathcal{S}(u) = \{n_1, n_2, \ldots, n_k\}$, are VNFs required by user u, and $\mathcal{S}(u) \subseteq \mathcal{N}, \forall u \in \mathcal{U}$. To model flow rate changes as a flow is processed by a VNF, we denote the ratio of the output and input flow rates by λ_n, $\forall n \in \mathcal{N}$. The communication latency between two nodes $i, j \in \mathcal{I}'$ is denoted by l_{ij}, where $\mathcal{I}' = \mathcal{I} \cup \mathcal{U} \cup \mathcal{U}'$. We assume $l_{ij} = 0$ when $i = j$, and $l_{ij} = l_{ji} > 0$ when $i \neq j$. In practice, latency is often dominated by distance-related propagation delay, and the delay introduced by reliability and congestion control protocols.

The data flow going through each service chain can be split and processed by different VNF instances of the same type, residing on possibly geo-dispersed nodes. Some VNF instances, such as DNS, can also be shared by multiple service chains, which can potentially reduce the overhead of creating and maintaining new VNF instances. Other functions, such as security-sensitive ones, do not permit sharing [17]. Each server node can host a large number of VNF instances in practice. We can assign the average overhead to each service chain proportional to its flow rate, so that the shareable and non-shareable VNFs can be unified in a single optimization model. Specifically, for a unit-rate flow processed by VNF n, we define φ_n^r as the average consumption of resource r, such that it consists of both the average resource consumed by processing the data flow and that for creating and maintaining VNF instances.

B. Service Chain Cost Minimization

A natural goal of a service provider is to minimize the overall cost of deploying a required set of service chains. We first consider the case of nonlinear resource cost function and linear latency cost function (the nonlinear case discussed in Sec. IV-B). Let $\mathscr{C}_j^r(\cdot)$ be a convex cost function of resource r on server node i, which is usually monotonically increasing in practice and may vary across servers. It is reasonable to assume that each type of VNF appears in a service chain at most once. The total resource cost at server nodes is:

$$J_R = \sum_{j\in\mathcal{I}} \Bigg[\mathscr{C}_j^{r_{out}} \Bigg(\sum_{u\in\mathcal{U}} \sum_{(m,n)\in\mathcal{L}(u)} \sum_{i\in\mathcal{I}^{u'}\setminus\{j\}} f_{ji}^{umn} \Bigg) + \sum_{r\in\mathcal{R}\setminus\{r_{out}\}} \mathscr{C}_j^r \Bigg(\sum_{u\in\mathcal{U}} \sum_{(m,n)\in\mathcal{L}(u)} \varphi_n^r \sum_{i\in\mathcal{I}^u} \overline{\delta}(i-j;r) f_{ij}^{umn} \Bigg) \Bigg], \tag{1}$$

where f_{ij}^{umn} is the flow of service chain u from VNF m on server node i to VNF n on server node j, and $\mathcal{I}^u = \mathcal{I} \cup \{u\}$, $\mathcal{I}^{u'} = \mathcal{I} \cup \{u'\}$. The function

$$\overline{\delta}(i-j;r) = \begin{cases} 0, & i = j, r = r_{in} \\ 1, & \text{otherwise} \end{cases}$$

is defined to indicate whether a flow consumes inbound bandwidth, since a flow from a node to itself requires no bandwidth resources.

In (1), the second term in the summation represents the cost of all types of resources except outbound bandwidth, which is in the first term. Interestingly, the first term is a function of only outgoing flows of server node j, while the second term is a function of incoming flows. This key observation is later exploited in our design of the distributed algorithm. For the second term, the cost of inbound bandwidth consumption can be included by setting $\varphi_n^r = 1$ for $r = r_{in}$.

Besides resource cost, the overall latency of a service chain is another QoS parameter for NFV users. The total latency cost is

$$J_L = \sum_{u\in\mathcal{U}} c^u \sum_{(m,n)\in\mathcal{L}(u)} \sum_{i\in\mathcal{I}^u} \sum_{j\in\mathcal{I}^{u'}} \frac{l_{ij} f_{ij}^{umn}}{\hat{\lambda}_m^u f^u}, \tag{2}$$

where $\hat{\lambda}_m^u = \lambda_{n_1}\lambda_{n_2}\cdots\lambda_m$ is the cumulative change ratio of flow rate along service chain $\mathcal{L}(u)$, and c^u is the unit cost of latency for user u. With slight abuse of notation, we use the same symbols for f_{ij}^{umn} and f^u, but note that only f_{ij}^{umn} is a decision variable. As shown in (2), the latency of each hop in a service chain is a weighted average of the latency along different paths.

The latency cost J_L is defined so to take the latencies of multiple concurrent paths of a service chain into consideration. For a single-path service chain, the overall latency degrades into the sum of per-hop latencies along the path, i.e., the end-to-end latency of the path. Although it would be simpler to upper bound the latency of each user by some fixed value in the form of a linear inequality constraint, we formulate it into a cost function to be minimized in order to cover more realistic scenarios, where high latency is undesirable but no hard constraint is set. Eq. (2) can be seen as both a real cost and a penalty term to avoid high latency.

The service chain cost minimization problem can now be formulated as:

$$\text{minimize} \quad J_R + J_L \tag{3a}$$

subject to:

$$\sum_{u\in\mathcal{U}} \sum_{(m,n)\in\mathcal{L}(u)} \varphi_n^r \sum_{i\in\mathcal{I}^u} \overline{\delta}(i-j;r) f_{ij}^{umn} \le C_j^r, \quad \forall j\in\mathcal{I}, r\in\mathcal{R}\backslash\{r_{out}\} \tag{3b}$$

$$\sum_{u\in\mathcal{U}} \sum_{(m,n)\in\mathcal{L}(u)} \sum_{k\in\mathcal{I}^{u'}\backslash\{j\}} f_{jk}^{umn} \le C_j^{r_{out}}, \forall j\in\mathcal{I} \tag{3c}$$

$$\sum_{k\in\mathcal{I}^{u'}} f_{jk}^{umn} = \lambda_m \sum_{i\in\mathcal{I}^u} f_{ij}^{ulm}, \quad \forall(l,m),(m,n)\in\mathcal{L}(u), u\in\mathcal{U}, j\in\mathcal{I} \tag{3d}$$

$$\sum_{j\in\mathcal{I}} f_{uk}^{uun} = f^u, \forall u\in\mathcal{U}, (u,n)\in\mathcal{L}(u) \tag{3e}$$

$$f_{ij}^{umn} \ge 0, \forall i\in\mathcal{I}^u, j\in\mathcal{I}^{u'}, (m,n)\in\mathcal{L}(u), u\in\mathcal{U} \tag{3f}$$

Constraints (3b) and (3c) capture resource capacity limits. Examples of resources include computational power, bandwidth, storage, and function-specific resources such as special devices (*e.g.*, GPGPU) and private data. For instance, if some function-specific resource required by a network function resides only on a few of the server nodes, we can simply set $C_j^r = 0$ for the others. Flow conservation is modeled by constraint (3d) and (3e). Assuming the server nodes are fully connected, the total number of decision variables is $O\left(|\mathcal{U}|\,|\mathcal{I}|^2\,|\mathcal{S}|_{max}\right)$, where $|\mathcal{S}|_{max} = \max_{u\in\mathcal{U}}|\mathcal{S}(u)|$.

IV. Distributed Algorithms for Service Chain Cost Minimization

A. Distributed Algorithm for Service Chain Provisioning

We now start to design a distributed ADMM-based algorithm for solving (3). We aim at parallelizing the optimization problem into $|\mathcal{I}|$ or $|\mathcal{I}|+|\mathcal{U}|$ subproblems, each corresponding to a server or source node. We first reformulate the overall resource cost function into:

$$J_R = \sum_{j\in\mathcal{I}} J_{R,j} = \sum_{j\in\mathcal{I}} \left[J_{R,j}^{in}\left(\boldsymbol{f}_j^{in}\right) + J_{R,j}^{out}\left(\boldsymbol{f}_j^{out}\right)\right], \tag{4a}$$

where

$$J_{R,j}^{in}\left(\boldsymbol{f}_j^{in}\right) = \sum_{r\in\mathcal{R}\backslash\{r_{out}\}} \mathscr{C}_j^r\Bigg(\sum_{u\in\mathcal{U}}\sum_{(m,n)\in\mathcal{L}(u)} \varphi_n^r \sum_{i\in\mathcal{I}^u} \overline{\delta}(i-j;r) f_{ij}^{umn}\Bigg), \tag{4b}$$

$$J_{R,j}^{out}\left(\boldsymbol{f}_j^{out}\right) = \mathscr{C}_j^{r_{out}}\Bigg(\sum_{u\in\mathcal{U}}\sum_{(m,n)\in\mathcal{L}(u)}\sum_{k\in\mathcal{I}^{u'}\backslash\{j\}} f_{jk}^{umn}\Bigg). \tag{4c}$$

Then we reformulate the total latency cost as

$$J_L = \sum_{j\in\mathcal{I}} J_{L,j} = \sum_{j\in\mathcal{I}} \left[J_{L,j}^{in}\left(\boldsymbol{f}_j^{in}\right) + J_{L,j}^{out}\left(\boldsymbol{f}_j^{out}\right)\right], \tag{5a}$$

where

$$J_{L,j}^{in}\left(\boldsymbol{f}_j^{in}\right) = \sum_{u\in\mathcal{U}} c^u \sum_{(m,n)\in\mathcal{L}(u)} \sum_{i\in\mathcal{I}^u} \frac{l_{ij} f_{ij}^{umn}}{\hat{\lambda}_m^u f^u}, \tag{5b}$$

$$J_{L,j}^{out}\left(\boldsymbol{f}_j^{out}\right) = \sum_{u\in\mathcal{U}} c^u \sum_{(m,u')\in\mathcal{L}(u)} \frac{l_{ju'} f_{ju'}^{umu'}}{\hat{\lambda}_m^u f^u}. \tag{5c}$$

For a specific server node j, $J_{R,j}^{in}$ and $J_{L,j}^{in}$ are determined by only the incoming flows, $\boldsymbol{f}_j^{in}$, whereas $J_{R,j}^{out}$ and $J_{L,j}^{out}$ are determined by only the outgoing flows, $\boldsymbol{f}_j^{out}$.

As shown in (4a) and (5a), J_R and J_L can be respectively decomposed into the sums of $J_{R,j}$ and $J_{L,j}$ over all server nodes. However, they cannot be optimized separately since each f_{ij}^{umn} is shared by two nodes. Similarly, constraint (3d) couples all server nodes.

We address these challenges using the method of auxiliary variables. First, observe that both $J_{R,j}$ and $J_{L,j}$ can be further decomposed into a function of incoming flows and a function of outgoing flows, as shown in (4a)-(5c). Second, it is apparent that the LHS of (3d) is the sum of outgoing flows of a node, whereas the RHS is the sum of incoming flows multiplied by a constant. We introduce a set of *auxiliary variables* $\boldsymbol{g}_j^{out} = \boldsymbol{f}_j^{out}, \forall j\in\mathcal{I}\cup\mathcal{U}$, such that the incoming and outgoing flows of a node are not shared by other nodes. We then derive an equivalent version of (3):

$$\text{minimize} \quad F(\boldsymbol{f}) + G(\boldsymbol{g}) \tag{6a}$$

subject to: constraints (3b), and

$$\sum_{u\in\mathcal{U}} \sum_{(m,n)\in\mathcal{L}(u)} \sum_{k\in\mathcal{I}^{u'}\backslash\{j\}} g_{jk}^{umn} \le C_j^{r_{out}}, \forall j\in\mathcal{I} \tag{6b}$$

$$\sum_{k\in\mathcal{I}^{u'}} g_{jk}^{umn} = \lambda_m \sum_{i\in\mathcal{I}^u} f_{ij}^{ulm}, \quad \forall(l,m),(m,n)\in\mathcal{L}(u), u\in\mathcal{U}, j\in\mathcal{I} \tag{6c}$$

$$\sum_{j\in\mathcal{I}} g_{uj}^{uun} = f^u, \forall u\in\mathcal{U}, (u,n)\in\mathcal{L}(u) \tag{6d}$$

$$g_{ij}^{umn} = f_{ij}^{umn}, \forall i \in \mathcal{I}^u, j \in \mathcal{I}, (m,n) \in \mathcal{L}(u), u \in \mathcal{U} \quad (6e)$$

$$f_{ij}^{umn} \geq 0, \forall i \in \mathcal{I}^u, j \in \mathcal{I}, (m,n) \in \mathcal{L}(u), u \in \mathcal{U} \quad (6f)$$

$$g_{ij}^{umn} \geq 0, \forall i \in \mathcal{I}^u, j \in \mathcal{I}^{u'}, (m,n) \in \mathcal{L}(u), u \in \mathcal{U} \quad (6g)$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are vectors of all f_{ij}^{umn}'s and g_{ij}^{umn}'s, and

$$F(\boldsymbol{f}) = \sum_{j \in \mathcal{I}} \left[J_{R,j}^{in}\left(\boldsymbol{f}_j^{in}\right) + J_{L,j}^{in}\left(\boldsymbol{f}_j^{in}\right) \right], \quad (7)$$

$$G(\boldsymbol{g}) = \sum_{j \in \mathcal{I}} \left[J_{R,j}^{out}\left(\boldsymbol{g}_j^{out}\right) + J_{L,j}^{out}\left(\boldsymbol{g}_j^{out}\right) \right]. \quad (8)$$

With the auxiliary variables, we not only reformulate (3a) into (6a) for separability over server nodes, but also decouple incoming and outgoing flows by replacing (3d) with (6c). Furthermore, the two sets of variables, $\boldsymbol{f}$ and $\boldsymbol{g}$, are coupled only by equality constraints (6c) and (6e), and problem (6) now fits the standard form of ADMM.

The augmented Lagrangian of problem (6) is

$$\begin{aligned} L_\rho(\boldsymbol{f},\boldsymbol{g},\boldsymbol{y}) &= F(\boldsymbol{f}) + G(\boldsymbol{g}) \\ &+ \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{S}(u)} \sum_{j \in \mathcal{I}} \left[y_{1,j}^{um} \left(\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} - \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \right) \right. \\ &\left. + \frac{\rho}{2} \left(\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} - \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \right)^2 \right] \\ &+ \sum_{u \in \mathcal{U}} \sum_{(m,n) \in \mathcal{L}(u)} \sum_{i \in \mathcal{I}^u} \sum_{j \in \mathcal{I}} \left[y_{2,ij}^{umn} \left(f_{ij}^{umn} - g_{ij}^{umn} \right) \right. \\ &\left. + \frac{\rho}{2} \left(f_{ij}^{umn} - g_{ij}^{umn} \right)^2 \right], \end{aligned} \quad (9)$$

where vector $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2)$ is the concatenation of dual variables $y_{1,j}^{um}$'s and $y_{2,ij}^{umn}$'s, and $(l,m),(m,n) \in \mathcal{L}(u)$ is abbreviated into $m \in \mathcal{S}(u)$. Here, $\rho > 0$ is the penalty parameter [11], which will also serve as the step size of update rules. We can now solve the cost minimization problem by alternatively updating $\boldsymbol{f}$ and $\boldsymbol{g}$, where each step is split into $O(|\mathcal{I}|)$ or $O(|\mathcal{I}| + |\mathcal{U}|)$ sub-problems.

The $\boldsymbol{f}$-update step requires solving:

$$\begin{aligned} \min_{\boldsymbol{f} \in \mathcal{C}_1} & F(\boldsymbol{f}) + \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{S}(u)} \sum_{j \in \mathcal{I}} \lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} \\ & \left[y_{1,j}^{um} + \frac{\rho}{2} \left(\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} - 2 \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \right) \right] \\ & + \sum_{u \in \mathcal{U}} \sum_{(m,n) \in \mathcal{L}(u)} \sum_{i \in \mathcal{I}^u} \sum_{j \in \mathcal{I}} f_{ij}^{umn} \left[y_{2,ij}^{umn} + \frac{\rho}{2} \left(f_{ij}^{umn} - 2 g_{ij}^{umn} \right) \right], \end{aligned} \quad (10)$$

where $\mathcal{C}_1$ is the convex polytope defined by constraints (3b) and (6f). By decomposing (10) over the server nodes, we derive the subproblem for each server node $j \in \mathcal{I}$:

$$\begin{aligned} \min_{\boldsymbol{f}_j^{in} \in \mathcal{C}_{1,j}} & J_{R,j}^{in}\left(\boldsymbol{f}_j^{in}\right) + J_{L,j}^{in}\left(\boldsymbol{f}_j^{in}\right) + \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{S}(u)} \lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} \\ & \left[y_{1,j}^{um} + \frac{\rho}{2} \left(\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} - 2 \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \right) \right] \\ & + \sum_{u \in \mathcal{U}} \sum_{(m,n) \in \mathcal{L}(u)} \sum_{i \in \mathcal{I}^u} f_{ij}^{umn} \left[y_{2,ij}^{umn} + \frac{\rho}{2} \left(f_{ij}^{umn} - 2 g_{ij}^{umn} \right) \right], \end{aligned} \quad (11)$$

where $\mathcal{C}_{1,j}$ is the convex polytope defined by part of (3b) and (6f) relevant to $\boldsymbol{f}_j^{in}$, the incoming flows of server j.

The $\boldsymbol{g}$-update step requires solving:

$$\begin{aligned} \min_{\boldsymbol{g} \in \mathcal{C}_2} & G(\boldsymbol{g}) + \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{S}(u)} \sum_{j \in \mathcal{I}} \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \\ & \left[- y_{1,j}^{um} + \frac{\rho}{2} \left(\sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} - 2\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} \right) \right] \\ & + \sum_{u \in \mathcal{U}} \sum_{(m,n) \in \mathcal{L}(u)} \sum_{j \in \mathcal{I}^u} \sum_{k \in \mathcal{I}} g_{jk}^{umn} \left[-y_{2,jk}^{umn} + \frac{\rho}{2} \left(g_{jk}^{umn} - 2 f_{jk}^{umn} \right) \right], \end{aligned} \quad (12)$$

where $\mathcal{C}_2$ is the convex polytope defined by (6b), (6d) and (6g). We changed the subscript of the last additive term from i,j in (9) to j,k for convenience. Decomposing (12) over the servers, we derive the subproblem for each server $j \in \mathcal{I}$,

$$\begin{aligned} \min_{\boldsymbol{g}_j^{out} \in \mathcal{C}_{2,j}} & J_{R,j}^{out}\left(\boldsymbol{g}_j^{out}\right) + J_{L,j}^{out}\left(\boldsymbol{g}_j^{out}\right) + \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{S}(u)} \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \\ & \left[- y_{1,j}^{um} + \frac{\rho}{2} \left(\sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} - 2\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} \right) \right] \\ & \sum_{u \in \mathcal{U}} \sum_{(m,n) \in \mathcal{L}(u)} \sum_{k \in \mathcal{I}} g_{jk}^{umn} \left[-y_{2,jk}^{umn} + \frac{\rho}{2} \left(g_{jk}^{umn} - 2 f_{jk}^{umn} \right) \right], \end{aligned} \quad (13)$$

where $\mathcal{C}_{2,j}$ is the convex polyhedron defined by part of the constraints (6b) and (6g) that is relevant to $\boldsymbol{g}_j^{out}$, the outgoing flows of server node j. However, different from the $\boldsymbol{f}$-update step, the combination of all subproblems (13) do not completely cover the original problem (12). The remaining problem to solve is:

$$\min_{\boldsymbol{g} \in \mathcal{C}_2'} \sum_{u \in \mathcal{U}} \sum_{(u,n) \in \mathcal{L}(u)} \sum_{k \in \mathcal{I}} g_{uk}^{uun} \left[-y_{2,uk}^{uun} + \frac{\rho}{2} \left(g_{uk}^{uun} - 2 f_{uk}^{uun} \right) \right] \quad (14)$$

where $\mathcal{C}_2'$ is the convex polytope defined by constraints (6d) and (6g). Fortunately, problem (14) is only relevant to the outgoing flows of source nodes $u \in \mathcal{U}$, and can be decomposed over the source nodes or service chains as

$$\min_{\boldsymbol{g}_u^{out} \in \mathcal{C}_{2,u}'} \sum_{(u,n) \in \mathcal{L}(u)} \sum_{k \in \mathcal{I}} g_{uk}^{uun} \left[-y_{2,uk}^{uun} + \frac{\rho}{2} \left(g_{uk}^{uun} - 2 f_{uk}^{uun} \right) \right] \quad (15)$$

where $\mathcal{C}_{2,u}'$ is the convex polytope defined by part of the constraints (6d) and (6g) relevant to $\boldsymbol{g}_u^{out}$. Subproblem (15) is simple enough to solve on each source node.

For the $(t+1)$-th iteration of the ADMM algorithm, we obtain $\boldsymbol{f}^{t+1}$ and $\boldsymbol{g}^{t+1}$ from $\boldsymbol{f}^t$, $\boldsymbol{g}^t$ and $\boldsymbol{y}^t$ in a sequential fashion, by solving subproblems (11), (13) and (15), and then we update the dual variables in $\boldsymbol{y}$ as

$$\begin{aligned} y_{1,j}^{um} &:= y_{1,j}^{um,k} + \rho \left(\lambda_m \sum_{i \in \mathcal{I}^u} f_{ij}^{ulm} - \sum_{k \in \mathcal{I}^{u'}} g_{jk}^{umn} \right), \\ & \forall j \in \mathcal{I}, (l,m),(m,n) \in \mathcal{L}(u), u \in \mathcal{U}, \end{aligned} \quad (16a)$$

and

$$\begin{aligned} y_{2,ij}^{umn} &:= y_{2,ij}^{umn} + \rho \left(f_{ij}^{umn} - g_{ij}^{umn} \right), \\ & \forall i \in \mathcal{I}^u, j \in \mathcal{I}, (m,n) \in \mathcal{L}(u), u \in \mathcal{U}. \end{aligned} \quad (16b)$$

Although the $\boldsymbol{f}$-update and $\boldsymbol{g}$-update steps are executed locally on each server or source node, they need to communicate the updated variables. For instance, to solve (11) in the $(t+1)$-th iteration, a server node j has to know the latest values of $\boldsymbol{g}_{ij}^{out}, \forall i \in \mathcal{I} \cup \mathcal{U}, i \neq j$, the outgoing flows of i sent to j, which are updated by other nodes. While as shown in (16), each server node $j \in \mathcal{I}$ only has to maintain a small set of dual variables related to j, so the $\boldsymbol{y}$-update step can be carried out locally, even without communicating about the dual variables maintained by other nodes.

B. Nonlinear Latency Cost Function

Our distributed algorithm design so far has assumed that the latency cost is a linear function, which may not always be the case in practice. For the more general nonlinear case, we can replace each coefficient c^u in (2) with a convex cost function $\mathscr{C}^u(\cdot)$, which can be interpreted as the latency tolerance curve of user u, such that

$$J_L = \sum_{u\in\mathcal{U}} \mathscr{C}^u\Bigg(\sum_{(m,n)\in\mathcal{L}(u)} \sum_{i\in\mathcal{I}^u} \sum_{j\in\mathcal{I}^{u'}} \frac{l_{ij} f_{ij}^{umn}}{\hat{\lambda}_m^u f^u} \Bigg). \tag{17}$$

Since the cost function $\mathscr{C}^u(\cdot)$ couples together all the flows of each service chain, J_L is no longer separable over the server nodes. Nevertheless, we can still decompose the problem over the source nodes or service chains, by introducing another set of auxiliary variables and the corresponding constraint,

$$h_i^u = \sum_{(m,n)\in\mathcal{L}(u)} \sum_{j\in\mathcal{I}^{u'}} \frac{l_{ij} g_{ij}^{umn}}{\hat{\lambda}_m^u f^u}, \forall i \in \mathcal{I}^u, u \in \mathcal{U}. \tag{18}$$

We then reformulate J_L as a function of h_i^u:

$$J_L = \sum_{u\in\mathcal{U}} \mathscr{C}^u\Bigg(\sum_{i\in\mathcal{I}^u} h_i^u \Bigg), \tag{19}$$

so that J_L is separable over $u \in \mathcal{U}$, and its parameters with different u are not directly coupled by any constraint. We can now update $(\boldsymbol{f}, \boldsymbol{h})$ in one step, and update $\boldsymbol{g}$ in the other, in a sequential fashion. Constraint (18) can be relaxed as we do to constraints (6c) and (6e). The separability of J_R and the relevant constraints remain intact, since constraint (18) does not couple g_{ij}^{umn} of different i. The rest of the algorithm for nonlinear J_L is similar to that presented in Sec. IV-A.

For nonconvex cost functions, ADMM can converge to a local minimum depending on the initial solution [11].

V. Performance Evaluation

In this section, we evaluate our NFV network model and the distributed optimization algorithm through simulation studies. We construct networks with the number of server nodes ranging from 20 to 50, and the number of service chains (hence the corresponding source and sink pairs) ranging from 5 to 15. These network nodes are uniformly deployed in a square-shape geographical area. The latency between each pair of nodes is assumed to be proportional to their distance. We setup 5 different types of VNFs, from which each service chain randomly choose 3 to 5. We further setup 5 different types of resources, including inbound and out bound bandwidth, and a possible type of function-specific resource. Other parameters, including flow rates, the unit cost and the available amount of resources are normally distributed.

We first investigate the convergence of the distributed algorithm. To improve the convergence rate of ADMM, we use the over-relaxation method proposed in [11], which introduces a relaxation parameter, $\alpha \in (0, 2)$. The stopping criterion we employ is based on measuring the residual errors in primal and dual feasibilities [11]. When the residual errors are smaller than pre-defined thresholds, the algorithm stops. We set $\rho = 0.4$, $\alpha = 1.8$, and track how the objective function value evolves over iterations in Fig. 1. The objective function rapidly reaches its optimum after about 100 iterations. After that, the algorithm further eliminates the residual errors in the primal and dual feasibilities before final termination.

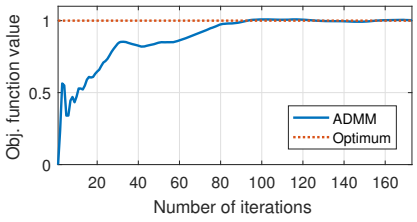


Fig. 1: $\rho = 0.4$, $\alpha = 1.8$. The ADMM-based algorithm terminates in 173 iterations.

To investigate the changes in J_R, J_L and J caused by larger service chain population, we set up a network with 50 server nodes. We then increase the number of service chains from 1 to 15, and minimize the total cost J respectively. As shown in Fig. 2, the growth trends of J_R and J_L are rather similar in this case; the total cost J increases super-linearly with the number of service chains, since the available resources become more and more scarce as demand escalates, and the algorithm is forced to utilize resources with high price and high latency.

Another interesting question is how the scarcity of function-specific resources affect the resource cost and the latency cost. We adopt the same network from the previous set of testing, and deploy 10 service chains. We then concentrate a specific type of resource from the 50 server nodes to only a few special server nodes and, at the same time, concentrate the demands of this resource from all 5 types of VNFs to only one of them, so that the available amount of the function-specific resource and its expected demand remain the same. The resource cost and the latency cost (when total cost is minimized) for different number of special server nodes are shown in Fig. 3.

In Fig. 3, as the number of special server nodes increases, the latency cost decreases faster than the resource cost when there are very few special server nodes, while the resource cost decreases slightly faster when the special server nodes are not scarce. This is because when the special server nodes are scarce, the service chains that require the function-specific resource are rather likely to make a long detour for that resource, which can be significantly alleviated by increasing the number of special server nodes.

We conduct the next set of simulations on two different networks, one with 20 server nodes and 5 service chains, the other with 50 server nodes and 10 service chains. We gradually increase the scale factor while keeping the total cost minimized. Fig. 4a and Fig. 4b show the results of the small network and the large network, respectively. The cost values are normalized by their initial values. For both networks, the total cost decreases monotonically and smoothly as resource capacities increase, while the resource costs and the latency costs fluctuate. The fluctuation is particularly evident in the small network, as the paths of the data flows tend to change dramatically when the number of server nodes is small.

It is also interesting to see what happens if we only minimize the resource cost or the latency cost. The results are presented in Fig. 5. When available resources are scarce, minimizing resource cost leads to higher total cost than when

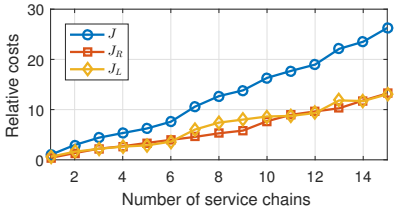


Fig. 2: Increase in costs due to increasing number of service chains.

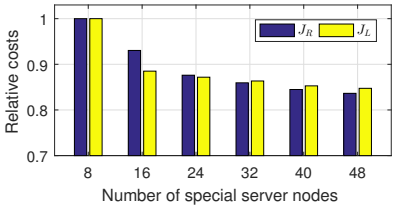
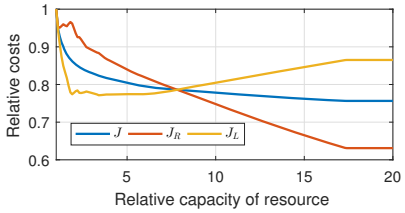
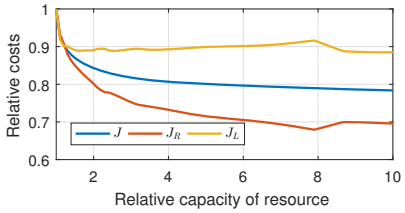


Fig. 3: Comparison of costs for different number of special server nodes.

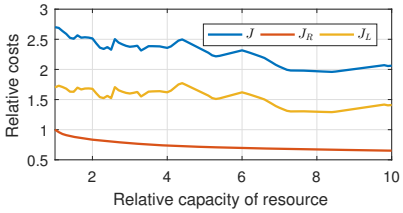


(a) Change in the costs of a small network.

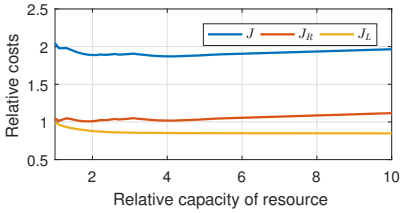


(b) Change in the costs of a large network.

Fig. 4: Change in costs as the capacities of resources increase.



(a) When only the resource cost is minimized.



(b) When only the latency cost is minimized.

Fig. 5: Change in costs as the capacities of resources increase.

minimizing latency cost; while minimizing the latency cost does not push the resource cost far from its minimum value. The fluctuations shown in Fig. 5a indicate dramatic changes in the paths of the data flows.

VI. Conclusion

In this work, we studied a fundamental problem arose in NFV service chain provisioning, that is jointly minimizing the overall resource cost and the end-to-end latency of service chains. We formulated the problem into nonlinear optimization, which was then transformed into an equivalent counterpart that fit the standard form of ADMM, and led to an efficient distributed algorithm. Interestingly, the technique we developed for the transformation can also be used to address other problems involving data flow routing. We evaluated the performance of the proposed model and the distributed algorithm for service chain cost minimization by extensive simulations.

Acknowledgments

This work was supported in part by grants from Hong Kong Research Grants Council under the contracts HKU 718513, 17204715, 17225516, 717812 and C7036-15G (CRF), grants from the National Natural Science Foundation of China under the contracts NSFC 61571335 and NSFC 61628209, and HKU URC Matching Funding.

References

[1] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[2] M. Chiosi, D. Clarke, P. Willis *et al.*, "NFV: An Introduction, Benefits, Enablers, Challenges & Call for Action," *ETSI White Paper*, 2012.

[3] ——, "NFV: Use Cases," *ETSI White Paper*, 2013.

[4] ——, "NFV: Architectural Framework," *ETSI White Paper*, 2013.

[5] ——, "NFV: Network Operator Perspectives on Industry Progress," *ETSI White Paper*, 2014.

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualization: Challenges and Opportunities for Innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[7] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near Optimal Placement of Virtual Network Functions," in *Proc. of IEEE INFOCOM*, 2015.

[8] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research Directions in Network Service Chaining," in *Proc. of IEEE SDN4FNS*, 2013.

[9] M. Chiosi, D. Clarke, P. Willis *et al.*, "NFV: Terminology for Main Concepts in NFV," *ETSI White Paper*, 2014.

[10] D. Gabay and B. Mercier, "A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] H. Xu and B. Li, "Joint Request Mapping and Response Routing for Geo-distributed Cloud Services," in *Proc. of IEEE INFOCOM*, 2013.

[13] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[14] H. Moens and F. De Turck, "VNF-P: A Model for Efficient Placement of Virtualized Network Functions," in *Proc. of IEEE CNSM*, 2014.

[15] O. Miksik, V. Vineet, P. Pérez, P. H. Torr, and F. Cesson Sévigné, "Distributed Non-convex ADMM-inference in Large-scale Random Fields," in *Proc. of BMVC*, 2014.

[16] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training Neural Networks Without Gradients: a Scalable ADMM Approach," *arXiv:1605.02026*, 2016.

[17] S. Gu, Z. Li, C. Wu, and C. Huang, "An Efficient Auction Mechanism for Service Chains in the NFV Market," in *Proc. of IEEE INFOCOM*, 2016.