Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining

Insun Jang, Dongeun Suh, Sangheon Pack, and György Dán

***Abstract*—In this paper, we consider the problem of optimal dynamic service function (SF) placement and flow routing in a service function chaining (SFC) enabled network. We formulate a multi-objective optimization problem to maximize the acceptable flow rate and to minimize the energy cost for multiple service chains. We transform the multi-objective optimization problem into a single-objective mixed integer linear programming (MILP) problem, and prove that the problem is NP-hard. We propose a polynomial time algorithm based on linear relaxation and rounding to approximate the optimal solution of the MILP. Extensive simulations are conducted to evaluate the effects of the energy budget, the network topology, and the amount of server resources on the acceptable flow rate. The results demonstrate that the proposed algorithm can achieve near-optimal performance and can significantly increase the acceptable flow rate and the service capacity compared to other algorithms under an energy cost budget.**

***Index Terms*—Service function chaining, acceptable flow rate, energy cost, flow-compensatory rounding based placement.**

I. Introduction

Today's networks rely on a variety of service functions (SFs) (often called middleboxes), such as firewalls and network address translators, for processing user traffic [1]. Depending on service requirements and policy the traffic of different users may have to be processed by different SFs in a particular order, which is often referred to as service function chaining (SFC) [2]. SFC is currently being discussed in the Internet Engineering Task Force (IETF) SFC working group (WG), including an SFC architecture [3] and various use cases (e.g., SFC in mobile networks [4]).

SFC today is based on proprietary network hardware appliances from independent vendors, which makes flexible and elastic resource management challenging. Network functions virtualization (NFV) [5] has been recently introduced to address this challenge. With NFV the SFs are virtualized (instead of hardware-based SFs or physical SFs), and an SF is placed (or instantiated) within a virtual machine (VM) [6], [7] on a commodity server while consuming server resources, e.g., CPU cycles and memory. Which commodity server a particular SF is placed, and how many instances of an SF are instantiated can be decided depending on the actual traffic volumes. NFV thus provides flexibility in allocating resources to SFs and in combining different instances of SFs into service chains (SCs), and can thus potentially reduce capital expenditure (CAPEX) and operational expenditure (OPEX) of network operators [8].

To fully leverage the flexibility of NFV enabled SFC, it is important to be able to route flows that use an SC on links and paths with sufficient bandwidth between the individual SFs. Such flow specific routing can be effectively achieved by the well-known concept of software defined networking (SDN) [9]. SDN decouples the control plane and the data plane, and allows a logically centralized controller in the control plane to define and install per-flow forwarding rules for flexible routing.

Given the flexibility provided by NFV and SDN, an important and challenging problem is to optimize the placement of SFs and the distribution of flows on network paths for SFC under link capacity and server resource constraints. This problem was considered in several recent works [10]–[20], with the common assumption that the amount of admitted flows (i.e., flow rate or throughput) for each SC is known. An equally important and challenging problem, which has received little attention so far, is to determine the service capacity of a network in terms of the acceptable flow rates for a set of SCs. Being able to determine the acceptable flow rate of individual SCs in an SFC-enabled network could allow operators to implement SC-based pricing and admission policies for maximizing revenues, and could be used by operators for link capacity and server resource dimensioning so as to maximize the flow rates for different SCs.

Motivated by these important use cases, in this paper we formulate the problem of maximizing the acceptable flow rate and minimizing the energy cost for multiple SCs in an NFV-enabled network as a multi-objective optimization problem. In the proposed optimization problem the placement of SFs (i.e., their locations and the number of SF instances) and flow distribution (i.e., the routes for SC flows) are jointly considered as decision variables, while the acceptable flow rate of individual SCs is determined based on pre-defined weight factors. We transform the multi-objective optimization problem into a single-objective mixed integer linear programming (MILP) problem by means of the ϵ-constraint method [21], and prove that the MILP problem is NP-hard. We then propose a rounding-based approximation, called flow-

A preliminary version of this paper was presented at IEEE Netsoft 2016, Seoul, Korea, June 2016 [13].

I. Jang, D. Suh, and S. Pack are with the School of Electrical Engineering, Korea University, Seoul, Korea. (e-mail:{zerantoss, fever1989, shpack}@korea.ac.kr, corresponding author: S. Pack).

G. Dán is with the School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden (e-mail: gyuri@ee.kth.se).

This work was supported in part by NRF of Korea Grant funded by the Korean Government (MSIT) (No. NRF-2014R1A2A1A12066986) and in part by IITP grant funded by the Korea government(MSIT) (No. 2017-0-00195).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2017.2760162, IEEE Journal on Selected Areas in Communications

compensatory rounding-based placement (FCRP), which runs in polynomial time and can be executed at a central entity (i.e., SDN controller). Simulation results demonstrate that the proposed algorithm can achieve near-optimal performance and significantly increase the acceptable flow rate compared to other algorithms under an energy cost budget.

The contributions of this paper are threefold. First, we study the *joint* optimization of maximizing the acceptable flow rate of each SC and minimizing the energy cost under resource capacity constraints in an NFV-enabled network. To this end, we determine the locations and the number of SF instances as well as the routes of SC flows, while the acceptable flow rate of each SC is determined by an SC-specific weight. Second, *weakly* Pareto-optimal solutions of the formulated optimization problem and solutions of the proposed algorithm capture an acceptable flow rate-energy cost curve, while through adjusting the SC weights, the solutions allow to explore the region of admissible flow rates as a function of the service capacity. Third, extensive simulation results show that the performance of the proposed algorithm is close to optimal while it can be easily implemented and operates in polynomial time, and give insight into the effect of various system parameters (e.g., energy cost budget) on the acceptable flow rates.

The rest of the paper is organized as follows. Section II discusses related work. The system model and the optimal resource allocation problem are described in Section III and Section IV, respectively. The proposed algorithm is described in Section V and extensive simulation results are presented in Section VI. Section VII concludes the paper.

II. Related work

Previous works on resource allocation for SFC either address routing for fixed SF placement [10]–[13] or routing for flexible SF placement [14]–[20], with consideration for various objectives.

In the case of fixed SF placement the problem is to determine the routes of SC flows for given SF locations. Kim *et al.* [10] propose algorithms for SC path selection so as to reduce the operation cost while Gushchin *et al.* [11] formulate a path optimization problem to minimize the number of routing rules when all SFs of an SC are consolidated in a single VM. Unlike ours, these works do not consider the processing order specified in the SCs, which is a fundamental requirement in SFC. Mijumbi *et al.* [12] formulate an online mapping and scheduling problem in which the SFs that each SC flow passes are scheduled over time so as to improve the revenue while reducing the cost in terms of computational resource utilization, not considering link capacity constraints. The paper proposes several greedy algorithms and a tabu search based heuristic algorithm. Jang *et al.* [13] formulate the problem of minimizing the network resource usage for a given SC flow as a linear programming (LP) problem which determines the routes of SC flows. Compared to [13], our work considers the joint optimization of SF placement and routing for SC flows as a multi-objective optimization problem, and allows simultaneous optimization of the acceptable flow rate and the energy cost.

In the case of flexible SF placement, the routes of SC flows and the locations of the SFs are determined simultaneously. Mehraghdam *et al.* [14] define SF graphs for each SC request, and formulates an optimization problem in which the SF graphs are mapped to a substrate network for optimizing link utilization, energy cost, and latency. Bari *et al.* [15] formulate the problem of minimizing OPEX and resource fragmentation. In the problem, directed graphs model SCs, which are mapped to a physical network using a heuristic algorithm based on the Viterbi algorithm. Baumgartner *et al.* [16] study a virtual mobile core network embedding problem to minimize the cost of link and node resources. The solution is based on embedding multiple subchains in the user and control planes of a given physical substrate network. These works can be considered as an extended version of the virtual network embedding problem [22], as they take into account the processing order in SFC. Different to our work, the maximization of the acceptable flow rates and the minimization of the energy cost are not taken into account in these works.

Ma *et al.* [17] present an SF placement problem with the objective of load balancing taking into account that SFs may reduce or expand flow rates, and propose a polynomial time algorithm. Moens *et al.* [18] propose an SF placement algorithm that considers services embedded in a VM (not individual SFs), and resources are assigned to VMs to minimize the number of active servers. Unlike our work, these works do not take into account the processing order specified in the SCs. Mohammadkhan *et al.* [19] formulate the problem of accommodating more flows in a domain by minimizing the maximum utilization of links and CPUs, and proposes several heuristics. This approach is different from ours, as it aims at maximizing the remaining resource capacity per link and CPU, instead of maximizing the acceptable flow rate. Li *et al.* [20] design a unified framework composed of an NFV orchestration system for SF placement and an SDN controller for routing. In the framework, a heuristic solution is proposed for minimizing the hop count between an ingress node and an egress node.

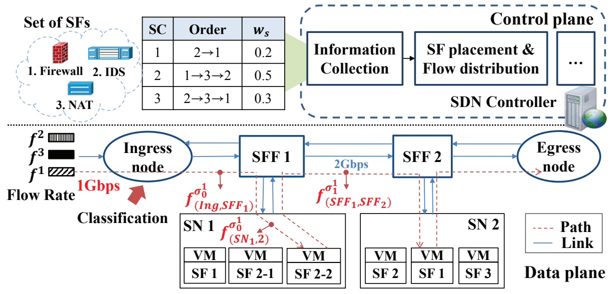
Compared to these previous works, our work allows to investigate the tradeoff between acceptable flow rate and energy cost for multiple SCs by determining both SF placement and routing, and makes it possible to characterize the service capacity of an SFC-enabled network in terms of acceptable flow rates.

III. System model

In this section, we present our model of an SFC enabled network, compatible with the IETF SFC architecture [3].

A. Network topology

We consider an SFC-enabled network, as shown in Figure 1, and model it as a directed graph $\mathcal{G} = (V, E)$. In accordance with the IETF SFC architecture [3], the set V of nodes consists of the set V_{ing} of ingress nodes, the set V_{egr} of egress nodes, the set V_{SN} of service nodes (SNs), and the set V_{SFF} of service function forwarders (SFFs), i.e., $V = V_{ing} \cup V_{egr} \cup V_{SN} \cup V_{SFF}$. For a directed edge $(i, j) \in E \subseteq V \times V$ we denote by $C_{(i,j)}$ its capacity, e.g., $C_{(SFF_1,SFF_2)} = 2$ Gbps in



SC	Order	w_s
1	2→1	0.2
2	1→3→2	0.5
3	2→3→1	0.3

Fig. 1. SFC-enabled network topology.

Figure 1. The ingress nodes and the egress nodes are entry and exit points of SC flows, respectively, and we make the natural assumption that the ingress nodes play the role of the classifier, as in [3], i.e., the ingress node performs the SFC encapsulation to identify a specific SF path (SFP). SFFs only forward flows to other SFFs or to SFs after checking the SFP information contained in the SFC encapsulation header. Note that according to [3] once processed by an SF placed on an SN, flows return to the SFF visited previously (e.g., SFF 1→SF 2-2→SFF 1 in Figure 1).

We consider that an SN is a commodity server that can host multiple VMs, with one SF instance per VM. Each SN is endowed with L types of resources (e.g., CPU and memory), and we denote by $A^i = \{a^i_1, ..., a^i_L\}$ the available resources of SN i. As discussed in the IETF SFC WG and shown in Figure 1, there is a central entity, the SDN controller, located in the control plane [23]. The SDN controller monitors and collects the status information in the SFC-enabled network (e.g., resources of nodes, link capacities, SC requests) and performs SF placement and flow distribution by solving the optimization problem presented in Section IV.

B. Service functions and service chains

We denote by $\mathcal{M} = \{1, ..., M\}$ the set of available SFs, and we denote by $R^m = \{r^m_1, ..., r^m_L\}$ the resource requirement of an instance of SF m (i.e., a VM template of SF m). We denote by C_m the maximum flow rate that SF m is able to serve, and refer to it as the processing capacity of an instance of SF m. An SF may change the rate of a flow that traverses it [24], for example, a firewall and a wide area network (WAN) optimizer can reduce the flow rate by dropping unpermitted flows and by compressing flows, respectively. To capture this, we denote by α_m the flow rate inflation factor of SF m [14], [17], e.g., $\alpha_m = 0.95$ implies that the flow rate decreases by 5% when a flow is processed by SF m.

A sequence of SFs forms an SC, and for an SC s we denote by T_s the number of SFs that it is composed of. An SC s can thus be represented by a vector $\sigma \in \mathcal{M}^{T_s}$, and we denote by σ^s_t the t^{th} SF in SC s. For example, in Figure 1 the first SF in SC 1 is SF 2, i.e., $\sigma^1_1 = 2$, and the length of SC 1 is $T_1 = 2$. A particular SF or an SF instance may appear in multiple SCs, possibly with different indices, e.g., in Figure 1 SF 2 is the first SF in SC 1, but is the third SF in SC 2. To ease presentation, we make use of the Kronecker delta $\delta_{\sigma^s_t\, m}$, e.g., in Figure 1 we have $\delta_{\sigma^1_1\, 2} = 1$ and $\delta_{\sigma^1_2\, 2} = 0$ because in SC 1 SF 2 is the first SF (not the second SF). Finally, we denote by $\mathcal{S} = \{1, ..., S\}$ the set of SCs.

For each $s \in \mathcal{S}$ we define the SC weight factor $w_s \in (0, 1]$, which is a system parameter, and can be used to prioritize SCs, e.g., depending on the number of customers using an SC.

C. SF placement decision variables

We allow multiple instances of an SF to be placed on a particular SN. We denote by $P_{i,m}$ the number of instances of SF m placed on SN i, and define the $M \times |V_{SN}|$ SF placement matrix $\mathbf{P} = (P_{i,m})$. As an example, in Figure 1 $P_{SN_1,SF_2} = 2$ because two instances of SF 2 are placed on SN 1. Furthermore, we define the set $\mathcal{P}$ of all possible SF placement matrices, which is a finite subset of all $M \times |V_{SN}|$ non-negative integer matrices, determined by the SNs' resource availabilities A^i and the SFs' resource requirements R^m.

D. Flow distribution decision variables

The placement of SFs together with the SF and link capacities determines the acceptable flow rate of the SCs. We denote by $f^s \in \mathbb{R}_{\geq 0}$ the acceptable flow rate of SC s, and we define the flow rate vector $\mathbf{F} = (f^1, ..., f^S)$. Furthermore, we denote by $\mathcal{F} \subset \mathbb{R}^S_{\geq 0}$ the set of acceptable flow rate vectors, and define the aggregate flow rate $\mathbb{F} = \sum_{s \in \mathcal{S}} f^s$.

To serve SCs at a flow rate higher than the capacity of individual links and SF instances, we allow the flow of an SC to be split among multiple SFPs. Since an SC flow is typically an aggregate of the traffic of multiple users, splitting can be done without the risk of packet re-ordering. To capture multiple SFPs for an SC s, we denote by $f^{\sigma^s_t}_{(i,j)}$ the flow rate of SC s that passes link (i, j) after being processed by SF σ^s_t. As a compact notation we define the order 3 tensor $\mathbf{F}_L = (f^{\sigma^s_t}_{(i,j)})$ for $(i, j) \in E, s \in \mathcal{S}, t \in \{0, T_s\}$, where σ^s_0 (i.e., $t = 0$) means that the flow of SC s is not yet processed by any SF and $\sigma^s_{T_s}$ means that the flow of SC s is processed by all SFs in SC s. Finally, we denote by $\mathcal{F}_L$ the set of possible $\mathbf{F}_L$ tensors. As an example, in Figure 1 there is an SFP for SC 1 with a flow rate of 1 Gbps (dashed line), thus $f^{\sigma^1_1}_{(SFF_1,SFF_2)} = 1$ Gbps since the flow traversing the link between SFF 1 and SFF 2 is already processed by the first SF in SC 1 (i.e., SF 2).

Similarly, for capturing the flow rates injected to SF instances, we denote by $f^{\sigma^s_t}_{(i,n)}$ the flow rate injected to the n^{th} instance of SF σ^s_{t+1} placed on SN i after being processed by SF σ^s_t. As a compact notation we use the order 4 tensor $\mathbf{F}_I = (f^{\sigma^s_t}_{(i,n)})$ for $i \in V_{SN}, s \in \mathcal{S}, t \in \{0, T_s - 1\}, n \in \{1, \max_{i,s,t} P_{i,\sigma^s_t}\}$. Finally, we denote by $\mathcal{F}_I$ the set of possible $\mathbf{F}_I$ tensors. As an example, in Figure 1 the flow of SC 1 is injected to the second instance (i.e., $n = 2$) of SF 2 (which is the first SF in SC 1) placed on SN 1, and thus $f^{\sigma^1_0}_{(SN_1,2)} = 1$ Gbps.

E. Normalized energy cost

We use the normalized energy cost to capture the energy efficiency of a placement. We make the reasonable assumption that the energy use is proportional to the amount of SN resources that are required to instantiate SFs [20], and define the resource-energy weight factor $e_l \in [0,1]$ to model the differing impact of different resources on the energy use (e.g., CPU utilization vs. memory use). We thus express the normalized energy cost as

$$\mathbb{E} = \sum_{i \in V_{SN}} \sum_{m \in \mathcal{M}} P_{i,m} \sum_{l=1}^{L} \frac{r_l^m}{\sum\limits_{i \in V_{SN}} a_l^i} e_l. \tag{1}$$

IV. Constraints and Problem Formulation

We are now ready to formulate the problem of finding an optimal placement of SF instances and corresponding SC flows that maximizes the flow rate of SCs and minimizes the energy cost. We start with introducing the flow conservation, capacity and SF order constraints, followed by the objective function.

A. SC flow conservation constraints

The first type of flow conservation constraint ensures that the sum of incoming flow rates to an SFF is equal to the sum of outgoing flow rates from the SFF,

$$\sum_{j \in V} f_{(j,i)}^{\sigma_t^s} = \sum_{j \in V} f_{(i,j)}^{\sigma_t^s}, \quad \forall s \in \mathcal{S}, \forall t \in \{0, T_s\}, \forall i \in V_{SFF}. \tag{2}$$

The second type of constraint ensures that incoming flow rates to an SN are the same as the outgoing flow rates from the SN, subject to the flow rate inflation factor,

$$f_{(j,i)}^{\sigma_{t-1}^s} \alpha_{\sigma_t^s} = f_{(i,j)}^{\sigma_t^s}, \quad \forall s \in \mathcal{S}, \forall t \in \{1, T_s\}, \forall j \in V_{SFF}, \forall i \in V_{SN}. \tag{3}$$

Since each ingress node is the starting point of aggregated flow rates, the third type of constraints requires that outgoing flows from an ingress node have not yet been processed by any SFs in an SC,

$$\sum_{t=1}^{T_s} \sum_{j \in V} f_{(i,j)}^{\sigma_t^s} = 0, \; \sum_{j \in V} f_{(i,j)}^{\sigma_0^s} = f^s, \quad \forall s \in \mathcal{S}, \forall i \in V_{ing}. \tag{4}$$

Similarly, incoming flows to an egress node should have been processed by all SFs in an SC,

$$\sum_{t=1}^{T_s} \sum_{j \in V} f_{(j,i)}^{\sigma_{t-1}^s} = 0, \quad \sum_{j \in V} f_{(j,i)}^{\sigma_{T_s}^s} = f^s \prod_{t=0}^{T_s} \alpha_{\sigma_t^s}, \quad \forall s \in \mathcal{S}, \forall i \in V_{egr}. \tag{5}$$

The fifth type of flow constraint ensures that incoming flows to an SN can be injected only to SF instances that are placed on the SN,

$$\sum_{j \in V_{SFF}} f_{(j,i)}^{\sigma_{t-1}^s} = \sum_{m \in \mathcal{M}} \sum_{n=1}^{P_{i,m}} \delta_{\sigma_t^s m} f_{(i,n)}^{\sigma_{t-1}^s}, \quad \forall s \in \mathcal{S}, \forall t \in \{1, T_s\}, \forall i \in V_{SN}. \tag{6}$$

The last type of flow constraint allows prioritization among competing SCs through the SC weight factor w_s. Our use of the weight factor is similar to generalized processor sharing [25], and ensures that a fraction w_s of the total flow rate is allocated to SC s,

$$f^s = w_s \sum_{s \in \mathcal{S}} f^s = w_s \mathbb{F}, \;\; \forall s \in \mathcal{S}. \tag{7}$$

B. Capacity constraints

The first type of capacity constraint ensures that link capacities are not exceeded,

$$\sum_{s \in \mathcal{S}} \sum_{t=0}^{T_s} f_{(i,j)}^{\sigma_t^s} \le C_{(i,j)}, \quad \forall (i,j) \in E. \tag{8}$$

The second type of capacity constraint ensures that the flow rate injected to an SF instance does not exceed the processing capacity of the SF instance,

$$\sum_{s \in \mathcal{S}} \sum_{t=1}^{T_s} \delta_{\sigma_t^s m} f_{(i,n)}^{\sigma_{t-1}^s} \le C_m, \quad \forall i \in V_{SN}, \forall m \in \mathcal{M}, \forall n \in \{1, P_{i,m}\}. \tag{9}$$

Finally, the third type of capacity constraint ensures that the capacity of an SN resource is not exceeded by the SF placement,

$$\sum_{m \in \mathcal{M}} r_l^m P_{i,m} \le a_l^i, \quad \forall l \in \{1, L\}, \forall i \in V_{SN}. \tag{10}$$

C. SC order constraints

To ensure that flows traverse SF instances in the order required by the SC, we require that flows processed by SF σ_t^s are directed to an SN with at least one instance of the subsequent SF, i.e., SF σ_{t+1}^s,

$$\sum_{j \in V_{SFF}} \sum_{s \in \mathcal{S}} \sum_{t=1}^{T_s} \delta_{\sigma_t^s m} f_{(j,i)}^{\sigma_{t-1}^s} \le P_{i,m} C_m, \quad \forall m \in \mathcal{M}, \forall i \in V_{SN}. \tag{11}$$

D. Decision variable constraints

Recall that $P_{i,m}$ takes non-negative integer values and $f_{(i,j)}^{\sigma_t^s}, f_{(i,n)}^{\sigma_t^s}, f^s$ take non-negative real values. We thus have

$$P_{i,m} \in \mathbb{Z}_{\ge 0}, \quad \forall i \in V_{SN}, \forall m \in \mathcal{M}, \tag{12}$$

$$0 \le f_{(i,j)}^{\sigma_t^s} \le f^s, \quad \forall (i,j) \in E, \forall s \in \mathcal{S}, \forall t \in \{0, T_s\}, \tag{13}$$

$$0 \le f_{(i,n)}^{\sigma_{t-1}^s} \le f_{(j,i)}^{\sigma_{t-1}^s}, \quad \forall i \in V_{SN}, \forall j \in V_{SFF}, \forall s \in \mathcal{S}, \forall t \in \{1, T_s\}, \forall n \in \{1, P_{i,\sigma_t^s}\}. \tag{14}$$

E. Joint Flow Maximization and Energy Minimization Problem

We are now ready to formulate the problem of maximizing the total acceptable flow rate and minimizing the energy cost in an SFC-enabled network as the following multi-objective optimization problem

$$\max_{\mathbf{F}\in\mathcal{F},\mathbf{F}_L\in\mathcal{F}_L,\mathbf{F}_I\in\mathcal{F}_I,\mathbf{P}\in\mathcal{P}} \quad (\mathbb{F}, -\mathbb{E}) \qquad (15)$$
$$\text{subject to} \quad (2)-(14).$$

In order to transform the above multi-objective problem into a single objective problem, we use the ϵ-constraint method [21], that is, the energy cost part of the objective function is converted into a constraint by introducing a threshold E_{Th} for the energy cost. The resulting Energy Cost Constrained Maximum Flow SF Placement (*EC-MaxF-SFP*) problem is then

$$\max_{\mathbf{F}\in\mathcal{F},\mathbf{F}_L\in\mathcal{F}_L,\mathbf{F}_I\in\mathcal{F}_I,\mathbf{P}\in\mathcal{P}} \quad \mathbb{F} \qquad (16)$$
$$\text{subject to} \quad (2)-(14), \quad \mathbb{E} \le E_{Th}.$$

It is easy to see that (16) is an MILP problem, and is computationally infeasible to solve in general, as shown by the following theorem.

Theorem 1: The EC-MaxF-SFP *problem is NP-hard.*

Proof: To show that the *EC-MaxF-SFP* problem is NP-hard, we provide a polynomial time reduction from the unbounded multiple knapsack (UMK) problem, which is known to be NP-hard [26].

The UMK problem considers a set of items $\mathcal{I} = \{1, ..., I\}$ with profit p_i and weight w_i, and a set of knapsacks $\mathcal{K} = \{1, ..., K\}$ with capacities c_k. The objective is to compute how many of each item to assign to each knapsack, denoted by $x_{ki}, \forall k \in \mathcal{K}, \forall i \in \mathcal{I}$, so as to maximize the total profit $\sum_{k=1}^{K}\sum_{i=1}^{I} p_i x_{ki}$) subject to knapsack capacity constraints $\sum_{i=1}^{I} w_i x_{ki} \le c_k, \forall k \in \mathcal{K}$.

We now show how to construct an instance of the *EC-MaxF-SFP* problem for solving an instance of the UMK problem. First, in the *EC-MaxF-SFP* problem all link capacities are set to infinite (i.e., big enough). This ensures that *EC-MaxF-SFP* only considers the acceptable flow rates at ingress nodes, regardless of flow conversation and SC order constraints. Second, the number of SN resource types is set to one, i.e., $L = 1$, and the energy cost threshold is set to 1.

Then, for each item i we create an SF m with resource requirement $r_1^m = w_i$, and capacity $C_m = p_i$. Furthermore, for each knapsack k we create an SN i with resource availability $a_1^i = c_k$. By doing so, the objective of *EC-MaxF-SFP* becomes the maximization of the total acceptable flow rate (the total profit of the UMK), and the resulting placement $P_{i,m}$ of SFs to SNs is the solution x_{ki} of the UMK. Consequently, the *EC-MaxF-SFP* problem is NP-hard. ■

Since the *EC-MaxF-SFP* problem is NP-hard, in what follows we propose a polynomial-time heuristic as a solution.

Algorithm 1 FCRP algorithm.

```
Input: $\mathcal{G}, A^i, R^m, C_m, \alpha_m, w_s, \sigma_t^s$
Output: $\mathbf{P}, \mathbf{F}, \mathbf{F}_L, \mathbf{F}_I$
    Step 1: Feasibility check
 1: if $E_{Th} \ge \sum_{m\in\mathcal{M}} \sum_{l\in[1,L]} \frac{r_l^m}{\sum_{i\in V_{SN}} a_l^i} e_l$ then  Go to Step 2
 2: else Stop Algorithm 1
 3: end if
    Step 2: LP relaxation
 4: Let $\mathbf{N_{max}} = \left(n_{i,m}^{max}\right)_{i\in V_{SN}, m\in\mathcal{M}}, n_{i,m}^{max} = \min_{l\in\{1,L\}} \left\lfloor \frac{a_l^i}{r_l^m} \right\rfloor$
 5: Solve EC-MaxF-SFP_LP($\mathbf{N_{max}}$) to obtain $\mathbf{P}^R$.
    Step 3: Placement adjustment
 6: for $m = 1$ to $M$ do
 7:     Let $i = \arg\max_{i\in V_{SN}} \{P_{i,m}^R\}$
 8:     if $P_{i,m}^R < 1$ and $\exists l \in [1, L]$, s.t. $a_l^i \ge r_l^m$, then
 9:         Set $P_{i,m}^R = 1$
10:     end if
11: end for
    Step 4: Flow rate loss evaluation
12: Let $\mathbf{P}^I = \left(P_{i,m}^I\right)_{i\in V_{SN}, m\in\mathcal{M}}, P_{i,m}^I = \lfloor P_{i,m}^R \rfloor$
13: Let $\mathbf{F}^d = \left(f_{i,m}^d\right) \in \mathbb{R}_{\ge 0}^{|V_{SN}|\times M}$
14: for $i = 1$ to $|V_{SN}|$ do
15:     for $m = 1$ to $M$ do
16:         Let $w_m = \sum_{s\in\mathcal{S}} \sum_{t=1}^{T_s} \delta_{\sigma_t^s m} w_s$
17:         Let $f_{i,m}^d = (P_{i,m}^R - P_{i,m}^I) C_m w_m / \sum_{m\in\mathcal{M}} w_m$
18:     end for
19: end for
    Step 5: Flow-compensatory rounding
20: while $\mathbf{F}^d > 0$ do
21:     Let $m' = \arg\max_{m\in\mathcal{M}} \left\{ \sum_{i\in V_{SN}} f_{i,m}^d \right\}$
22:     Let $i' = \arg\max_{i\in V_{SN}} \{f_{i,m'}^d\}$
23:     Set $P_{i',m'}^I = P_{i',m'}^I + 1, \ f_{i',m'}^d = 0$
24:     if $\exists l \in \{1, L\}$ s.t. $a_l^{i'} < \sum_{m\in\mathcal{M}} P_{i',m}^I C_m$, or $\mathbb{E} > E_{Th}$ then
25:         Set $P_{i',m'}^I = P_{i',m'}^I - 1, \ f_{i',m'}^d = 0$
26:     end if
27: end while
    Step 6: Optimal path creation
28: Solve MaxF_LP $\left(\mathbf{P}^I\right)$ to obtain $\mathbf{F}, \mathbf{F}_L, \mathbf{F}_I$.
```

V. Flow-Compensatory Rounding-based Placement Algorithm

The proposed flow-compensatory rounding-based placement (FCRP) algorithm approaches the maximum acceptable flow rate by iteratively reducing the flow rate loss compared to an optimal fractional solution. In what follows we explain the operation of *FCRP* and analyze its computational complexity.

A. Algorithm Description

Algorithm 1 shows the pseudo-code of *FCRP*. The algorithm consists of six steps. The first step is to verify the feasibility of *EC-MaxF-SFP*. If the problem is feasible, *FCRP* creates an LP relaxation of *EC-MaxF-SFP* referred to as *EC-MaxF-SFP_LP*. In other words, integer variables (i.e., $P_{i,m}$) in *EC-MaxF-SFP* are replaced by real variables, which makes *EC-MaxF-SFP_LP* an LP problem. Therefore, the relaxed problem can be solved in polynomial time via well-known algorithms (e.g., interior point method [27]) implemented in various LP solvers (e.g., IBM ILOG CPLEX). As the third step, *FCRP* adjusts the real-valued SF placement solution to ensure that each SF has at least one instance. In the fourth step, it rounds the adjusted real-valued solution for the SF placement, and evaluates the flow rate loss due to rounding. In the fifth step, it iteratively adjusts the rounded SF placement to reduce the flow rate loss. Finally, for the computed SF placement it computes the optimal flow vectors by solving the LP problem (referred to as *MaxF_LP*) obtained from *EC-MaxF-SFP* with the fixed SF placement.

Step 1) Feasibility check: The feasibility check verifies that the given energy cost threshold (i.e., E_{Th}) is sufficient for instantiating at least one instance per each SF. The algorithm terminates if this is not the case.

Step 2) LP relaxation: Note that the index n in $f_{i,n}^{\sigma_t^s}$ refers to the n^{th} instance of an SF in problem (16), and its maximum value is determined by $P_{i,m}$, which is the number of instances of SF m on SN i, and is the integer variable of the MILP.

In order to make an LP relaxation possible, we compute the maximum number $n_{i,m}^{max}$ of SF instances that can be placed on SN i by rounding down the ratio of the resource capacity of the SN and the resource requirement of an SF instance (line 4). We then temporarily create fictitious instances of each SF up to the maximum number on each SN. Therefore, the relaxed *EC-MaxF-SFP_LP* problem includes flow variables $f_{i,n}^{\sigma_t^s}$ for all $1 \leq n \leq n_{i,m}^{max}$, and constraints $f_{i,n}^{\sigma_t^s} = 0$ for $n > P_{i,m}$. We then solve the *EC-MaxF-SFP_LP* problem, which provides a real-valued SF placement matrix $\mathbf{P}^R$ (line 5).

Step 3) Placement adjustment: In order to avoid solutions in which a particular SF is not placed on any SN, which may occur due to the rounding in the next step, if there is an SF m for which the highest value of $P_{i,m}^R$ is less than 1, we round it up to 1 (lines 6-11). This ensures that at least one instance of SF m, and thus constraint (7) can be satisfied.

Step 4) Flow rate loss evaluation: To create an integer placement matrix we round down $\mathbf{P}^R$ to the nearest integer, resulting in $\mathbf{P}^I$ (line 12). Rounding leads to a decrease of the aggregate processing capacities of SF instances, which results in a decrease (loss) of flow rate. We thus compute the flow rate loss matrix $\mathbf{F}^d = (f_{i,m}^d)$, whose entries capture the flow rate loss for SN i and SF m (lines 13-19). Recall that the flow rate for each SC has a different weight, which is taken into account when computing a weight factor w_m for each SF (line 16). Given the SF weights, we can use the decrease of the processing capacity (i.e., $(P_{i,m}^R - P_{i,m}^I) * C_m$) combined with the share of SFs (i.e., $w_m / \sum_{m \in \mathcal{M}} w_m$) (line 17) to compute the decrease of flow rate.

Step 5) Flow-compensatory rounding: In the fifth step we iteratively adjust the integer placement matrix $\mathbf{P}^I$. Rounding is performed once for every node-SF pair in $\mathbf{F}^d$, i.e., a total of $M \times |V_{SN}|$ times (lines 20-27), which guarantees that the running time of the algorithm is bounded. *FCRP* first finds an SF (m') with maximum loss rate among all SNs (line 21), and then selects an SN (i') that has the largest loss rate for the observed SF (line 22). The reason why we choose the SF first is to balance the number of instances among SFs belonging to the same SC, as doing so helps to avoid an SF becoming a bottleneck for an SC. To compensate the flow rate loss, $P_{i',m'}^I$ is incremented for SN i' and SF m', and the corresponding flow rate loss is set to 0 (line 25). Note that the rounding procedure should not lead to a violation of resource capacity of any SN or the energy cost threshold E_{Th} (lines 24-26).

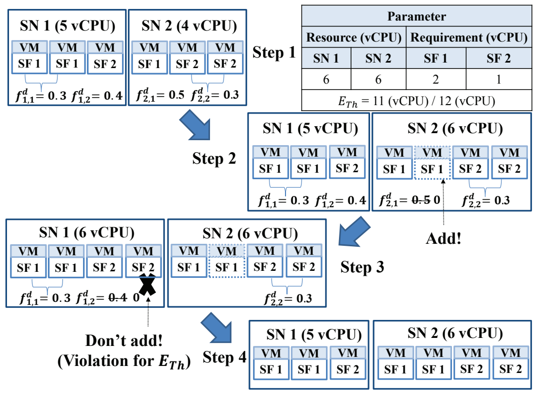


Fig. 2. Example of FCRP algorithm.

Step 6) Optimal path creation: Given the integer SF placement matrix $\mathbf{P}^I$, the last step consists of solving the *MaxF_LP* problem to find the maximum acceptable flow rate and the optimal paths, i.e., $\mathbf{F}, \mathbf{F}_L, \mathbf{F}_I$ (line 28). Note that the *MaxF_LP* problem is obtained from the *EC-MaxF-SFP* by removing constraints (10), (12), and the energy cost constraint, together with the given $\mathbf{P}^I$.

B. Flow-Compensatory Rounding Example

We illustrate the operation of *FCRP* through an example shown in Figure 2. The example assumes that the SF placement and the flow rate losses have been computed, as described in Steps $1-4$. In the example two instances of SF 1 and one instance of SF 2 are placed on SN 1, and one instance of SF 1 and two instances of SF 2 are placed on SN 2. Also, available resources of SN 1 and SN 2 are equally set to 6 virtual CPUs (vCPUs) and the number of vCPUs required by an instance of SF 1 and SF 2 are 2 and 1, respectively. Therefore, SN 1 and SN 2 are consuming 5 vCPUs and 4 vCPUs, respectively.

In the example, the algorithm starts with the given SF placement at Step 1 in Figure 2. Then we assume that the corresponding flow rate loss for SF 1 is 0.8, given by the sum of $f_{1,1}^d = 0.3$ in SN 1 and $f_{2,1}^d = 0.5$ in SN 2 while the flow rate loss for SF 2 is 0.7, given by the sum of $f_{1,2}^d = 0.4$ in SN 1 and $f_{2,2}^d = 0.3$ in SN 2. Therefore, as described in Step 5 of Algorithm 1, SF 1 is selected first, since its flow rate loss is larger than that for SF 2. Next, SN 2 is chosen for SF 1, because the flow rate loss in SN 2 for SF 1 (0.5) is larger than that in SN 1 for SF 1 (0.3). As an adjustment one instance of SF 1 is added on SN 2 for accepting a higher flow rate, as shown at Step 2 in Figure 2. After Step 2 in Figure 2, the algorithm performs the same procedure.

The following iteration first selects SF 2, since its flow rate loss (0.7) is larger than that for SF 1 (0.3). SN 1 is then

TABLE I
DEFAULT PARAMETERS IN SIMULATIONS.

Description	Value
Number of SFFs / SNs / links	10 / 6 / 42
E_{Th}	0.5
vCPU requirements of SFs	$(1, 1, 2, 2, 4)$ [29]
Processing capacities of SFs	$(0.5, 0.5, 1, 1, 2.5)$ (Gbps) [29]
Flow rate inflation factors of SFs	$(0.8, 1, 1.2, 1, 1)$
Set of SC weights	$(0.4, 0.3, 0.2, 0.1)$

selected owing to the larger flow rate loss (0.4), as shown at Step 3 in Figure 2. Unfortunately, adding one instance of SF 2 exceeds the threshold for the energy cost (i.e., $11/12$), which makes this adjustment infeasible. The rounding procedure terminates eventually, after it iterates through the remaining pairs, whose final placement is shown at Step 4 in Figure 2.

C. Complexity Analysis

In Algorithm 1, the **for** loop in lines 15-18 iterates M times, and the number of iterations of the **for** loop at lines 14-19 is $|V_{SN}|$. Thus the number of iterations in the nested **for** loop is $M \times |V_{SN}|$, which includes the **for** loop iterating M times in lines 6-11. In addition, the maximum number of iterations of the **while** loop in lines 20-27 is the size of matrix $\mathbf{F}^d$, i.e., $M \times |V_{SN}|$. Algorithm 1 solves two LP problems, i.e., *EC-MaxF-SFP_LP* and *MaxF_LP* problems. The former *EC-MaxF-SFP_LP* problem includes all variables defined in the *EC-MaxF-SFP* problem. Their sizes are S for $\mathbf{F}$, $\sum_{s\in\mathcal{S}} T_s \times |E|$ for $\mathbf{F}_L$, $\sum_{s\in\mathcal{S}} T_s \times |V_{SN}| \times |\mathbf{N_{max}}|$ for $\mathbf{F}_I$, and $|V_{SN}| \times M$ for $\mathbf{P}$, respectively. We denote by T_C the sum of the sizes. Note that the complexity of solving the latter *MaxF_LP* problem is lower than that of solving the former problem since the SF placement matrix $\mathbf{P}$ is excluded as variable. Since solving the LP problem can be done in polynomial time, the complexity of *FCRP* is polynomial in T_C. In particular, if an interior point method with complexity $O(N^{3.5})$ is used for solving the LP problem, the complexity of *FCRP* becomes $O(T_C^{3.5})$. Given recent developments in SDN/NFV technologies and high-performance controllers for WANs, the *FCRP* algorithm could be executed in real time for medium sized network topologies.

VI. PERFORMANCE EVALUATION

We use simulations to provide insight into the performance of the proposed *FCRP* algorithm and the acceptable flow rates. To evaluate the performance of *FCRP*, we consider an SFC-enabled network on the Abilene WAN topology (consisting of 12 nodes) in Internet2 [28]. The nodes in the network topology are mapped to SFFs, an ingress node, and an egress node. The ingress node and the egress node are chosen to be at least two hops away. We connect 6 SNs to the mapped SFFs, and choose the link capacities from the discrete uniform distribution (DUnif) on $\{5, 10\}$ (Gbps). Since usually CPU capacity is the most limited resource and the CPU usage dominates the energy cost, we only consider CPU resources in the simulations, and choose the number of vCPUs of each SN from the DUnif on $\{20, 25, 30\}$. For this network topology, we consider 5 SFs,

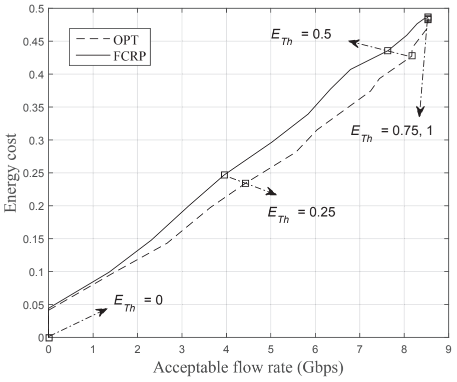


Fig. 3. Pareto frontier obtained based on the optimal solution and using FCRP.

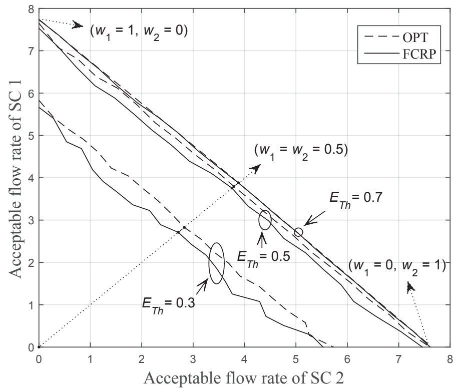


Fig. 4. Region of admissible flow rates for $S = 2$.

and we create 4 SCs by randomly choosing subsets of 3 or 4 SFs. The parameters are summarized in Table I. As a baseline for comparison we consider two placement algorithms and two routing algorithms.

- **Randomized placement (RP) algorithm**: The algorithm randomly places each SF instance on an SN, subject to node capacity constraints.
- **SF-aware placement (SFAP) algorithm**: The algorithm sorts the SFs in ascending order of their flow inflation factors α_m, and sorts the SNs in ascending order of their hop-distance from the ingress node. It then places SFs on SNs in order.
- **Multi path routing (MPR) algorithm**: Given the locations and the number of SF instances, this algorithm determines multiple routing paths per SC by solving the *MaxF_LP* problem.
- **Single path routing (SPR) algorithm**: Given the SF placement, the algorithm chooses a single routing path between each pair of SNs that can accept the maximum flow in an SC. The single path can be computed using well-known algorithms for solving the maximum flow problem (e.g., Edmonds-Karp algorithm [30])

Given the two placement algorithms and the two routing algorithms, we compare *FCRP* to 5 algorithms, namely, *FCRP-SPR*, *RP-MPR*, *RP-SPR*, *SFAP-MPR*, and *SFAP-SPR*. Note that *FCRP-SPR* supports *SPR* instead of *MPR* applied in *FCRP*. For fair comparison with *FCRP*, E_{Th} for the other algorithms is set to the energy cost obtained from *FCRP*, and the initial number of SF instances is commonly determined according to w_m in Algorithm 1. As a result, the energy costs obtained using all algorithms are fairly similar. We thus show simulation results only for the (aggregate) acceptable flow rate, with the exception of Figure 3. All simulation results shown are averaged over 200 simulation runs, and 95% confidence intervals are shown.

A. Pareto frontier and Service capacity

We first compare the optimal solution obtained by solving the *EC-MaxF-SFP* problem to the solution obtained by *FCRP*, by plotting the Pareto frontier, i.e., the achievable set of combinations of energy cost and acceptable flow rate that cannot be improved upon without deteriorating one of the two. To obtain the Pareto frontier, we solved the *EC-MaxF-SFP* problem for various values of E_{Th} between 0 and 1, each solution providing us a *weakly* Pareto-optimal solution [21], and plotted the resulting energy cost and acceptable flow rate.

Figure 3 shows the Pareto frontier of the flow rate (x-axis) and the energy cost (y-axis) obtained by solving *EC-MaxF-SFP* (denoted by *OPT*) and obtained by *FCRP*. To interpret the figure, observe that points above the curves are not Pareto efficient and points below the curves are not achievable. The figure shows that *FCRP* achieves near-optimal performance, as the relative difference to *OPT* is less than 13.3% in terms of the flow rate and 7.5% in terms of the energy cost. We can also observe that the trade-off between energy cost and flow rate is approximately linear. At the same time, it is important to note that the actual energy cost does not increase proportional to the energy cost threshold E_{Th} for high values of the threshold (e.g., for $E_{Th} = 0.75$ the actual energy cost is below 0.5), which shows that the link capacity constraints limit the acceptable flow rates above a certain amount of SN resources. Overall, the figure shows that the proposed *FCRP* algorithm allows to explore nearly Pareto efficient combinations of energy cost and acceptable flow rate, and can thus be used for long term capacity planning and for characterizing the trade-off between flow rate and energy cost.

Figure 4 shows the regions of acceptable flow rates of 2 SCs for *FCRP* and *OPT*, respectively with 3 different values of E_{Th} where the region is obtained by varying w_1 between 0 and 1. The figure shows the capacity region of the network in terms of acceptable flow rates, and confirms that the capacity region is convex and can be explored using *FCRP*.

B. Effect of energy cost threshold E_{Th}

Figure 5 shows the acceptable flow rates as a function of the energy cost threshold E_{Th} for six algorithm combinations. The figure shows that the acceptable flow rate increases with E_{Th} with a decreasing marginal gain, as the amount of usable SN resources increases, confirming our previous observation for Figure 3. At around $E_{Th} = 0.7$ the link capacity starts to limit the acceptable flow rates for all six algorithms, but as shown in the figure, *FCRP* achieves up to 35% and 95% higher flow rates than *RP* and *SFAP*, respectively.

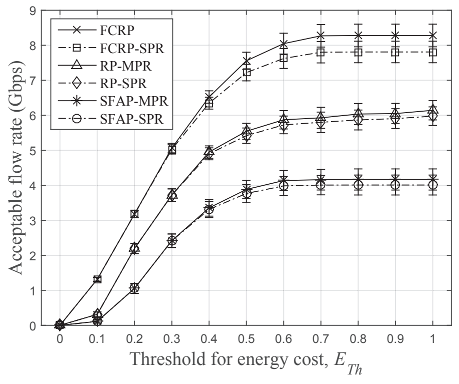


Fig. 5. Effect of energy cost threshold E_{Th} on acceptable flow rate.

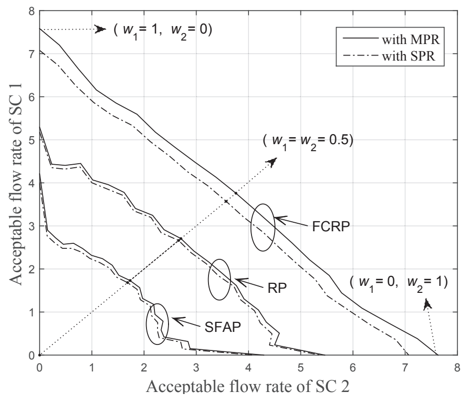


Fig. 6. Region of acceptable flow rates ($S = 2$).

It is interesting to note that even though *RP* places SF instances at random without consideration of the processing order of the SCs and the locations of SF instances, it performs better than *SFAP*, which places SF instances based on their flow rate inflation factor. The reason is that RP can avoid low capacity links on the paths of SC flows by evenly distributing the SFs in the network.

Comparing single and multi-path routing, Figure 5 shows that multipath routing has little impact on the flow rate on the considered network topology, as most flows traverse a single path (that can accept the largest flow rate) between SNs. We can thus conclude that it is the placement of SFs, respecting SC order and resource availability, that are most important for maximizing the acceptable flow rate.

Figure 6 shows the acceptable flow rate regions for all six considered algorithms for $S = 2$, obtained by varying w_1 between 0 and 1. We observe that not only does *FCRP* have the largest acceptable flow rate region, but it is also the only algorithm that results in a convex flow rate region. The reason for the irregular (non-convex) shape of the curves for the other two algorithms is that they do not account for the processing order of the SCs for SF placement.

C. Effect of network topology

In order to assess the impact of the network topology on the performance of the algorithms next we consider the Geant topology (23 nodes) [28] and a 4-ary Fat-tree (36 nodes) [31],

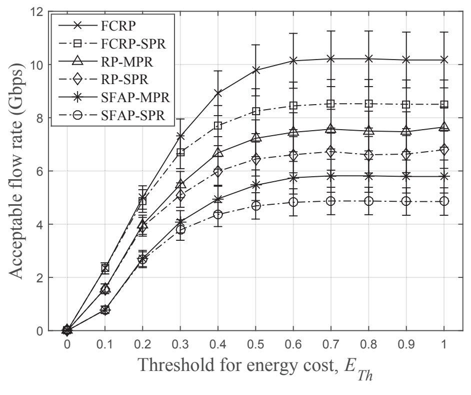


Fig. 7. Acceptable flow rate vs energy cost threshold for the Geant topology.

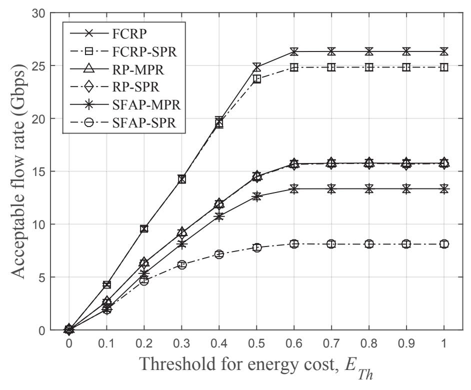


Fig. 8. Acceptable flow rate vs energy cost threshold for the Fat tree topology.

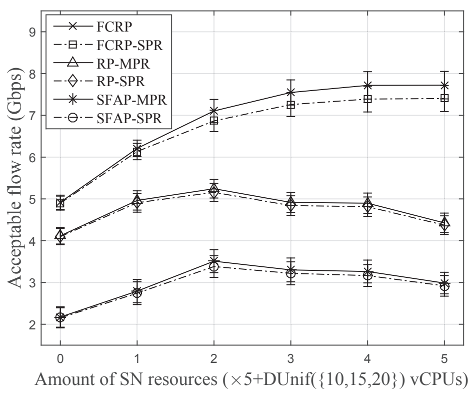


Fig. 9. Effect of SN resource.

which is a well-known data center network topology, besides the Abilene network. We added 12 SNs to the Geant network topology, as it is more complex and larger than the Abilene network. For the Fat-tree we mapped core switches to an ingress node and an egress node, host servers to SNs, and remaining switches to SFFs.

Figure 7 and Figure 8 show the flow rate as a function of E_{Th} for the Geant and the Fat tree topologies and for the six algorithms, respectively, and show that *FCRP* performs best overall. At the same time, Figure 7 highlights the importance of optimal multi-path routing on the Geant topology, which allows for more paths between SNs than the Abilene topology. Similarly, Figure 8 shows that the performance gain of *FCRP* is more significant on the Fat-tree topology than on Abilene and on Geant, which is due to that the *RP* and *SFAP* algorithms do not consider the SF order, and thus there is a high probability that flows traverse the same links between SNs, even though the Fat-tree topology has multiple paths between SNs. This shows that on a regular topology it is essential to use a placement algorithm that considers the order of SFs.

D. Effect of SN resource availability

Figure 9 shows the flow rate as a function of the amount of SN resources for the six algorithms for the Abilene topology. The figure shows that the flow rates obtained using *FCRP* and *FCRP-SPR* increase with the amount of SN resources with a decreasing marginal gain, which is due to the saturation of the link capacities. As *FCRP* captures both the computational and link capacity constraints, it could be used for optimal dimensioning of the SN capacity for given link capacities and network topology, as well as for link dimensioning for given SN capacities.

Interestingly, contrary to expectations and to *FCRP*, the flow rates of *RP* and *SFAP* do not increase monotonically as a function of the available SN resources. This is because they may place SF instances on SNs connected to low capacity links, and as the amount of SN resources increases, flows' competition for link capacity reduces the acceptable flow rates. This observation again highlights the importance of capturing computational and communication constraints together with SC order requirements for the optimization of SF placement.

VII. Conclusion

In this paper we considered the placement of SFs and corresponding routing with the aim of maximizing the acceptable flow rate and minimizing the energy cost in an SFC-enabled network. We transformed the multi-objective optimization problem into an MILP problem with a single-objective, and proposed a rounding-based heuristic algorithm, *FCRP*, with low computational complexity. Simulation results demonstrate that *FCRP* achieves near-optimal performance and can significantly increase the acceptable flow rate under an energy cost budget compared with other algorithms. The presented results allow network operators to compute the acceptable flow rates and the service capacity when multiple SCs are needed, as a function of various networks and SF parameters. As part of our future work we plan to address the problem of resource allocation for a sequence of SC requests subject to reconfiguration constraints in the framework of sequential decision making problems. As another extension, we also plan to investigate resource optimization issues in container-based SFC.

References

[1] J. Sherry and S. Ratnasamy, "A Survey of Enterprise Middlebox Deployments," EECS Dept., Univ. of California, Berkeley, Tech. Rep., UCB/EECS-2012-24, Feb. 2012.

[2] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," *IEEE Comm. Mag.*, vol. 55, no. 2, pp. 216–223, Feb. 2017.

[3] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015.

[4] W. Haeffner, J. Napper, M. Stiemerling, D. Lopez, and J. Uttaro, "Service Function Chaining Use Cases in Mobile Networks," draft-ietf-sfc-use-case-mobility-07, Oct. 2016.

[5] ETSI NFV, "Network Functions Virtualisation (NFV); Architectural Framework," ETSI GS NFV 002 v1.2.1, Dec. 2014.

[6] 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. USENIX NSDI*, Apr. 2014.

[7] J. Hwang, K. K. Ramakrishnan, and T. Wood, "NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms," *IEEE Trans. on Netw. and Serv. Mgmt*, vol. 12, no. 1, pp. 34–47, Mar. 2015.

[8] J. Soares, C. Goncalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, "Toward a Telco Cloud Environment for Service Functions," *IEEE Comm. Mag.*, vol. 53, no. 2, pp. 98–106, Feb. 2015.

[9] Bruno Astuto A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, Q3 2014.

[10] T. Kim, S. Kim, K. Lee, and S. Park, "A QoS Assured Network Service Chaining Algorithm in Network Function Virtualization Architecture," in *Proc. IEEE/ACM CCGrid*, May 2015.

[11] A. Gushchin, A. Walid, and A. Tang, "Scalable Routing in SDN-enabled Networks with Consolidated Middleboxes," in *Proc. ACM SIGCOMM Workshops on HotMiddlebox*, Aug. 2015.

[12] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and Evaluation of Algorithms for Mapping and Scheduling of Virtual Network Functions," in *Proc. IEEE NetSoft*, Apr. 2015.

[13] I. Jang, S. Choo, M. Kim, S. Pack, and M.-K.Shin, "Optimal Network Resource Utilization in Service Function Chaining," in *Proc. IEEE NetSoft*, June 2016.

[14] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," in *Proc. IEEE CloudNet*, Oct. 2014.

[15] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," *IEEE Trans. on Netw. and Serv. Mgmt*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[16] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile Core Network Vitualization: A Model for Combined Virtual Core Network Function Placement and Topology Optimization," in *Proc. IEEE NetSoft*, Apr. 2015.

[17] W. Ma, C. Medina, and D. Pan, "Traffic-Aware Placement of NFV Middleboxes," in *Proc. IEEE Globecom*, Dec. 2015.

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

[19] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood, "Virtual Function Placement and Traffic Steering in Flexible and Dynamic Software Defined Networks," in *Proc. IEEE LANMAN*, Apr. 2015.

[20] Y. Li, F. Zheng, M. Chen, and D. Jin, "A Unified Control and Optimization Framework for Dynamical Service Chaining in Software Defined NFV System," *IEEE Wireless Commun.*, vol. 22, no. 6, pp. 15–23, Dec. 2015.

[21] M. Ehrgott, "Multicriteria Optimization," 2nd ed., Springer, 2010.

[22] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Q4 2013.

[23] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components & Requirements," draft-ietf-sfc-control-plane-08, October 2016.

[24] Z. A. Qazi, C.-C.Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in *Proc. ACM SIGCOMM*, Aug. 2013.

[25] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[26] H. Kellerer, U. Pferschy, and Pisinger, "Knapsack Problems," Springer-Verlag, 2004.

[27] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.

[28] SNDlib, [Online]. Available:http://sndlib.zib.de/home.action.

[29] Cisco, "Cisco Cloud Services Router 1000V 3.14 Series Data Sheet,"

[30] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problem," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972.

[31] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. ACM SIGCOMM*, Aug. 2008.

Insun Jang received the B.S. and Ph.D. degrees from Korea University, Seoul, Korea, in 2011 and 2017, respectively. He is working at LG Electronics. His research interests include content delivery networks, mobile cloud networking, SDN/NFV, Future Internet, and vehicular networks.

Dongeun Suh received the B.S. degrees from Korea University, Seoul, Korea, in 2012. He is currently an Ph.D. course student in School of Electrical Engineering, Korea University, Seoul, Korea. His research interests include HTTP-based adaptive multimedia streaming, delay-tolerant networking, and SDN.

Sangheon Pack received the B.S. and Ph.D. degrees from Seoul National University, Seoul, Korea, in 2000 and 2005, respectively, both in computer engineering. In 2007, he joined the faculty of Korea University, Seoul, Korea, where he is currently a Professor in the School of Electrical Engineering. He was the recipient of KICS (Korean Institute of Communications and Information Sciences) Haedong Young Scholar Award 2013, IEEE ComSoc APB Outstanding Young Researcher Award in 2009, and LG Yonam Foundation Overseas Research Professor Program in 2012. He was a publication co-chair of IEEE INFOCOM 2014/ACM MobiHoc 2015, a co-chair of IEEE VTC 2010-Fall transportation track, a co-chair of IEEE WCSP 2013 wireless networking symposium, a TPC vice-chair of ICOIN 2013, and a publicity co-chair of IEEE SECON 2012. He is an editor of Journal of Communications Networks (JCN) and a senior member of the IEEE. His research interests include Future Internet, SDN/ICN/DTN, mobility management, mobile cloud networking, multimedia networking, and vehicular networks.

György Dán is an associate professor at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, a Fulbright research scholar at University of Illinois at Urbana-Champaign in 2012-2013, and an invited professor at EPFL in 2014-2015. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security in power systems.