Low-latency Service Chaining with Predefined NSH-based Multipath across Multiple Datacenters

Yao-Chun Wang, Ren-Hung Hwang, *Senior Member, IEEE*, and Ying-Dar Lin, *Fellow, IEEE*

***Abstract*—Service Function Chaining (SFC) provides a method of forwarding traffic flows through one or more service functions (SFs). For service providers, chaining SFs across multiple datacenters to deliver end-to-end services not only provides better utilization of computing resources of datacenters, but also achieves scalability and fault tolerance. However, most telecommunication applications are sensitive to latency, which tends to degrade due to both virtualization and the long distances among datacenters. In this paper, we extend the network service header (NSH) protocol and propose a multipath chaining with the partially-ordered NSH (MCPON) mechanism to achieve low-latency, partially-ordered service function chaining. MCPON adopts a proactive multipath installation for commonly-used service function paths (SFP, a sequence of requisite SFs) to eliminate reactive path decision delays and to reduce end-to-end service latency. To increase multipath diversity for better load balancing, we modify the original NSH encapsulation design so that the multiple paths selected for an SFP are not limited to having the same execution orders of some non-order-constrained SFs. MCPON also utilizes an entry-saving forwarding table design which enables forwarding entries to be shared among different SFC requests. Our evaluations show that proactive *k*-path computation for an SFC of length *l* at a scale of *n* SFFs saves time complexity of $O(kl^3n^3)$, and multipath service chaining reduces latency by 33–68% compared to single-path service chaining in our simulation scenarios.**

***Index Terms*—Multipath, Network Service Header, Service function chaining**

I. Introduction

SERVICE function chaining (SFC) [1], [2] provides a method of forwarding traffic flows through one or more service functions (SFs) in some specific order for the delivery of end-to-end services. To increase the flexibility and reduce capital expenditure (CAPEX) and operational expenditure (OPEX) of service deployments, network function virtualization (NFV) [3] was developed to transform much hardware network equipment (e.g., routers, firewalls, and load balancers) into virtual network functions (VNFs), which can be consolidated onto commodity servers and switches in datacenters. Since NFV creates a very dynamic network environment driven by customers requesting on-demand services and operators aiming to efficiently manage the performance of services, Software Defined Networking (SDN) [4], [5] plays an important role in the orchestration of NFV infrastructure resources (e.g., physical and virtual switching) by offering comprehensive network monitoring and dynamic provisioning of network connectivity, bandwidth, and security policy. Several studies [6], [7] highlight the benefits of SDN and NFV on providing efficient network resource allocation and QoS improvement in different types of networks. A good example is the Central Office Re-architected as a Datacenter (CORD) [8], [9] platform which integrates NFV, SDN and cloud-native architecture design to transform telco central offices into agile edge datacenters, for network operators to deliver innovative services with great user experience. Service providers like AT&T and Verizon are already supporting CORD.

For service providers, chaining VNFs or SFs across multiple datacenters (CORDs, core datacenters, or clouds) [10], [11] to deliver end-to-end services not only provides better utilization of the computing resources of datacenters but also achieves scalability and fault tolerance. However, most telecommunication applications are sensitive to service latency, which tends to degrade due to both virtualization and the long distances among datacenters [11]. In this paper, we focus on delivering low-latency service function chaining in a software-defined multi-datacenter environment and propose an MCPON (multipath chaining with partially-ordered NSH) mechanism, which adopts proactive multipath installation for SFC requests with consideration of the order of SFs, the network latencies, and the processing latencies of SFs.

A. Service Function Chaining with SDN and Network Service Header

To serve an incoming service chain request in service function chaining (SFC) architecture [1], a classifier (or service classification function) selects a suitable service function path (SFP, a sequence of SFs) that traverses all requisite SFs in a specific order (fixed-ordered SFP), and then encapsulates the request flow with network service header (NSH) [12] which

Manuscript received Sep 11, 2021; revised Mar 12, 2022; accepted Jul 15, 2022.

Y. C. Wang is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan (e-mail: scott0612@cht.com.tw).

R. H. Hwang is with the College of Artificial Intelligence, National Yang Ming Chiao Tung University, Hsinchu, Taiwan (e-mail: rhhwang@cs.ccu.edu.tw).

Y. D. Lin is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan (e-mail: ydlin@cs.nctu.edu.tw).

TABLE I
ACRONYMS IN SFC ARCHITECTURE AND NSH

Acronyms	Terms	Defined in
SFC	Service Function Chain	RFC 7665
SF	Service Function	
SFF	Service Function Forwarder	
SFP	Service Function Path	
NSH	Network Service Header	RFC 8300
SPI	Service Path Identifier	
SI	Service Index	

carries the information of the SFP, and finally sends the encapsulated flow to service function forwarders (SFFs) for service delivery. An NSH is mainly composed of a Service Path Header, which contains a Service Path Identifier (SPI), and a Service Index (SI). The SPI uniquely identifies an SFP, and the SI provides a location within the SFP. An SFF can determine the next hop for the requisite SF according to the SPI/SI values. In addition, the SI will be decremented by 1 by the serving SF after performing the required service. Table I lists some common acronyms defined in SFC architecture [1] and NSH [12].

With SDN, the network intelligence is decoupled from the forwarding plane and centralized in SDN controllers that control several devices with a global view of a network. OpenFlow [13], [14] is the premier standardized interface between SDN controllers and switches. By setting the default policy of SDN (or OpenFlow) switches, the first packet of a new fixed-ordered or partially-ordered (some SFs can be executed in a flexible order) service chain request can be directed to an SDN controller, which reactively decides a suitable SFP and installs the best single forwarding path, and then redirects the packet to a classifier (i.e., an edge SDN switch which executes packet encapsulation), for starting the end-to-end service delivery. In view of this, our proposal aims to reduce forwarding path decision delays (including computation and configuration) and end-to-end service latency.

B. The Proposed Method: MCPON

In order to achieve low-latency service function chaining, we present an MCPON (multipath chaining with partially-ordered NSH) mechanism, which reduces forwarding path decision delays and end-to-end service latency for each new fixed/partially-ordered SFC request.

To reduce a forwarding path decision delay, MCPON proactively computes forwarding paths and installs corresponding forwarding entries in SFFs for commonly-used SFPs. The commonly-used SFPs are some predefined popular lists of SFs, e.g., Firewall, IDS, and DPI. To serve a new incoming service chain request, an SDN controller only needs to install an entry in the classifier (for matching characteristics of the request flow) once the requisite SFs (SFP) of the new request has been decided so that the classifier can encapsulate packets according to the entry, and forward the encapsulated packets to the starting SFF for the delivery along a predefined forwarding path. If an SFC request arrives and it does not belong to any of these commonly-used SFPs, then MCA will add the new SFP to commonly-used SFPs and reactively compute forwarding paths. In this case, path decision delay is inevitable for the first time.

To further reduce end-to-end service latency, MCPON uses multipath routing. In addition, multipath routing provides network load balancing and fault tolerance in the presence of network link failures, which may affect service availability and user Service Level Agreement (SLA) [15]. With this approach, SFC requests, which have the same requisite SFs and the same starting SFF, will be matched by the same forwarding entries but may be forwarded to different paths.

In addition, to increase multipath diversity (i.e., options for path selection) for better load balancing, the multiple paths we selected for an SFP may have different execution orders of some non-order-constrained SFs. For a partially-ordered SFC request, the actual execution order of SFs may depend on the path to be traversed; however, a partially-ordered SFP is not supported by the original NSH design. In light of this, MCPON adopts a modified NSH which allows the SFC encapsulation to have different execution orders of SFs from the order defined by the service path identifier (SPI).

Our proposal considers partially-ordered SFC for the following reasons. First, partially-ordered SFC is explicitly defined as one of the SFCs to be supported in RFC 7498. It is also very common in real-world applications. For example, the order of executing 'URL Filtering' and 'Email Spam Filtering' in a security service chain does not affect the results. Second, fixed-ordered SFC is a special case of partially-ordered SFC, which means MCPON can also handle fixed-ordered SFC requests. MCPON specifically provides the following features:

1) Elimination of path decision delay with proactive path installation and modified NSH

The packets of a partially-ordered SFC request are encapsulated with modified NSH (or partially-ordered NSH) and forwarded along one of the predefined multiple paths. To support partially-ordered SFP (flexible execution order of SFs in SFP), we modify the usage of the 24 bits SPI and 8 bits SI in NSH.

2) Low end-to-end service latency with weighted multipath

We propose a multipath chaining algorithm (MCA) to compute weights of multiple forwarding paths with consideration of the order of SFs, the network latencies and the processing latencies of SFs. The path weight is regarded as the probability of selecting the path.

3) Entry-saving forwarding table design

To forward packets encapsulated with the proposed partially-ordered NSH, we design a forwarding table in SDN-enabled SFF, termed the multipath chaining table (MCT), to consult the modified SPI/SI values in order to determine the next hops. Since SDN switches rely on limited Ternary Content-Addressable Memory (TCAM) [16] to store forwarding entries,

scalability is a major concern for service providers. To reduce forwarding entry consumption, MCT also enables forwarding entries to be shared among different SFC requests. The reduction of TCAM requirement may also help to reduce network CAPEX.

In the evaluation of MCPON, we prepare a set of SFCs of different lengths and measure the reactive path computation times (which can be saved by proactive path installation) for the set of SFCs at different scales of an inter-datacenter network. We also measure the consumption of forwarding entries that are proactively installed to SFFs by MCPON for the set of SFCs (in the cases of using different numbers of multiple paths), and then we estimate the corresponding latency performance of these SFCs with random traffic injection in a simulation environment. We further compare the forwarding entry consumption with per-request forwarding (no share of forwarding entries between SFCs).

In summary, we propose a proactive multipath chaining mechanism with partially-ordered NSH that achieves low-latency partially-ordered SFC, by reducing forwarding path decision delay and end-to-end service latency. We also design an entry-saving forwarding table that enables forwarding entries to be shared among different SFC requests. The main contribution of this work is that we designed a partially-ordered NSH along with MCT that can forward the SFC encapsulations along multiple paths which may have different execution order of SFs.

The remainder of this paper is organized as follows: Section II discusses related works; Section III describes the problem; Section IV illustrates solution details; Section V shows the experimental results and related observations, and Section VI contains the conclusions and future work.

II. Related Work

Service function placement ([11], [17], [18], [19]), service chaining ([10], [20], [21], [22], [23], [24]) and dynamic scaling ([25], [26]) are key challenges [27] of providing service function chaining. The placement problem aims to determine the optimal SF locations to meet service requirements while optimizing resource utilization. The chaining problem aims to find the optimal SFC path according to different service requirements, such as end-to-end latency, generated cost, and energy consumption. And the scaling problem investigates ways to flexibly deploy SF instances in order to cope with the changing network workloads while improving energy efficiency.

In this work, we focus on the service chaining problem and propose a comprehensive solution, including path decision and packet forwarding, to achieve low-latency service function chaining.

Table II gives a summary of related works. For path decision, several path-finding solutions [10], [20], [21], [22] focus on finding the best path for each request flow. Vertex-centric distributed resource orchestration [10] finds all feasible mappings of an SFC (which can be further pruned to obtain the best SFC to satisfy the constraint) in multi-domain networks without replication of global state information, via message exchange between vertices (physical nodes capable of invoking a subset of SFs). A heuristic algorithm, QoS-Guaranteed SFC Outsourcing algorithm (QGSO) has been developed [20] to find the cost-efficient path based on Hidden Markov Model (HMM), with consideration of the order of VNFs in SFC, the QoS requirements, and the diverse pricing schemes of VNFs of different cloud providers. Adaptive Service Routing (ASR) algorithm [21] is a novel method that transforms the network representation to a layered graph that considers processing steps and allows the use of conventional shortest path algorithms (Dijkstra's algorithm) to determine the best path for an SFC. Energy-aware routing (EAR) [22] uses the breadth-first search (BFS) algorithm to find the best path that jointly optimizes the server energy and bandwidth costs for dynamic SFC deployment. As noted above, MCPON proactively installs forwarding paths to eliminate path decision delays. To avoid traffic congestion caused by mapping all SFC requests that have the same requisite SFs to a single path, MCPON adopts weighted multipath routing for network load balancing.

For packet encapsulation and forwarding table design, the NSH offers a common and standards-based header for service chaining to all network and service nodes. NSH/RFC 8300 [12] gives an example to illustrate weighing SFs (for load

TABLE II
Related Works

	Category	**Order of SFs in SFC request**	**Path decision**	**Encapsulation and Forwarding**
Vertex-centric distributed resource orchestration [10]	Path-finding algorithm	Partially-ordered	Single path	N/A
QGSO [20]		Fixed-ordered	Single path	
ASR [21]		Partially-ordered	Single path	
EAR [22]		Fixed-ordered	Single path	
NSH [12]	Encapsulation and forwarding table design	N/A	N/A	Fixed-ordered SFP, multipath
CRT-Chain [23]				Fixed-ordered SFP, single path
KeySFC [24]				Fixed-ordered SFP, single path
MCPON	Comprehensive solution	Partially-ordered	Weighted multiple paths	Partially-ordered SFP, multipath

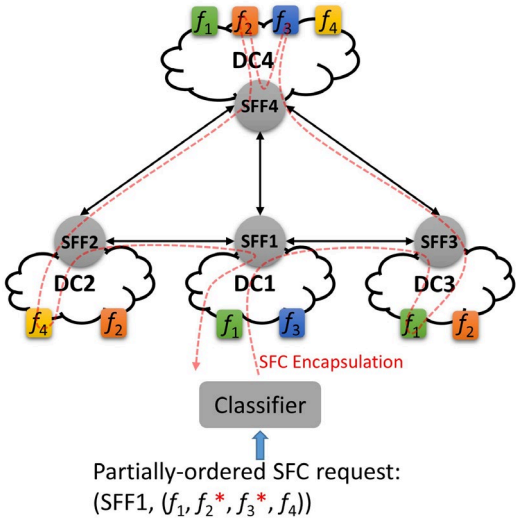


Fig. 1. A scenario of service function chaining across multiple datacenters.

distribution or redundancy); in this case, the table lookup (occurring on an SFF) may return more than one possible next hop within an SFP for a given SF. MCPON adopts NSH-Based service chaining and further extends the idea of weighing SFs to weighed multipath, enabling SFC requests that have the same requisite SFs to have the same header and to traverse multiple possible paths (which may have different execution order of some flexible-ordered SFs). To do so, MCPON has to modify the original NSH and forwarding table design for supporting multipath forwarding with partially-ordered SFP. CRT-Chain [23] is a service chain forwarding protocol that leverages the Chinese Remainder Theorem (CRT) to compress the forwarding information into small labels (which can replace the 32-bit service path header of the legacy NSH). With CRT-Chain, an SFF only needs to conduct simple modular arithmetic to extract the forwarding rules directly from CRT-Chain's labels attached in the header and requires only constant forwarding entries, regardless of the number of SFC requests. Similarly, KeySFC [24] also performs efficient forwarding using the residue numeral system (RNS). However, CRT-Chain and KeySFC are unable to support multipath forwarding since the table lookup with the modular arithmetic can return only one next hop.

Compared with existing approaches, our proposal involves the design of path decision, packet encapsulation, and a forwarding table to support multipath service chaining to reduce end-to-end service latency. A comparison of latency performance with single-path service chaining will be evaluated in Section V.

III. Problem Statement

A scenario of service function chaining across multiple datacenters is shown in Fig. 1, and the notations used in the problem description are given in Table III. We assume that a service provider owns a set of NFV-enabled datacenters (C), which are interconnected via a set of overlay links (L) between a set of SFFs (S) to provide SFC in a single SFC-enabled domain [1]. Each datacenter c_i ($c_i \in C$) has a border SFF s_i ($s_i \in S$), and a pair of SFFs (s_i, s_j) is connected by an overlay link $link_{ij}$ ($link_{ij} \in L$).

TABLE III
Notations

Categories	Notations	Descriptions
Topology	$G=(S, L)$	The network topology with SFFs S and overlay links L.
DC	C	The set of datacenters.
	c_i	The i-th datacenters, $c_i \in C$.
SFF	s_i	The SFF of c_i, $s_i \in S$. Each datacenter has a border SFF.
Link	$link_{ij}$	The overlay link from s_i to s_j. $s_i, s_j \in S$; $link_{ij} \in L$; $i \neq j$.
	t_link_{ij}	The measured network latency of $link_{ij}$.
SF	F	The set of SFs.
	f_i	The i-th SF, $f_i \in F$.
	f_{ij}	The j-th instance of f_i. Each datacenter has at most one instance of f_i.
	t_f_{ij}	The measured processing latency of f_{ij}, which consists of the roundtrip network latency between f_{ij} and local SFF and the processing latency at f_{ij}.
	$l(f_{ij})$	The location of f_{ij} (i.e., the datacenter in which f_{ij} is located).
	F^*	The set of non-order-constrained SFs, $F^* \subseteq F$.
	f_i^*	$f_i = f_i^*$ if $f_i \in F^*$. The execution order of cascading non-order-constrained SFs in a request can be interchanged.
Request	$r_n = (s_i, (rf_1, rf_2, \ldots, rf_k))$	s_i: The SFF of the nearest edge datacenter.
		$(rf_1, rf_2, \ldots, rf_k)$: The SFP, i.e., an ordered sequence of k requisite SFs.
Multipath	$path_{ni}$	The i-th path for request r_n.
	t_path_{ni}	The latency of $path_{ni}$.
	w_{ni}	The weight (probability of selection) of $path_{ni}$.
	$avg\_t\_path_n$	The weighted average path latency for request r_n.

The service provider provides a set of SFs (F), and each kind of SF f_i ($f_i \in F$) may have multiple instances (f_{ij}) which are allocated in multiple datacenters. The location of f_{ij} (i.e., the datacenter in which f_{ij} is located) is denoted by $l(f_{ij})$. We assume that each datacenter has at most one instance of f_i.

Each SFC request r_n from the user will be directed to the SFF of the nearest edge datacenter (by a classifier) for service and can be served by more than one datacenter. The edge datacenter can handle the received SFC request on its own or make use of the SF resources in other datacenters (i.e., chaining SFs in a single datacenter or across multiple datacenters), depending on its capabilities. We assume that the forwarding path of an SFC request starts and terminates (be decapsulated) at the same SFF (i.e., the SFF of the nearest edge datacenter).

The SFC request is denoted by $r_n = (s_i, (rf_1, rf_2, \ldots, rf_k))$,

which contains the information of the nearest SFF s_i and an SFP, i.e., an ordered sequence of k requisite SFs $(rf_1, rf_2, \ldots, rf_k)$, $rf_{i=1\sim k} \in F$. We assume that some SFs are not order-constrained, which means that the execution order of cascading non-order-constrained SFs could be interchanged. The set of non-order-constrained SFs is denoted by F^* ($F^* \subseteq F, f_i = f_i^*$ if $f_i \in F^*$). For example, for an SFC request which has an SFP $(f_1, f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7)$, the execution order of f_2^*, f_3^* can be swapped, and so do f_5^*, f_6^*.

Our proposal aims to reduce the forwarding path decision delay and minimize the end-to-end service latency, so as to deliver low-latency service function chaining in a multi-datacenter environment. As noted above, the path decision delay could be eliminated by the proactive installation of forwarding paths. Thus, the objective of MCPON is to determine the forwarding plane design along with the forwarding entries to be proactively installed for each commonly-used SFP, which minimizes the average service latency of SFC requests that have the same starting SFF and the same SFP.

The service latency of an SFC request includes network latencies of overlay links and processing latencies of SFs. The network latency of link $link_{ij}$ is denoted by t_link_{ij}, and the processing latency of SF f_{ij} is denoted by t_f_{ij}, which represents the roundtrip latency between f_{ij} and local SFF (i.e., the SFF of the datacenter in which f_{ij} is located).

More precisely, the objective of MCPON can be described as follows:

given

1. the network topology $G = (S, L)$ of a set of datacenters C,
2. the measured latencies of all network links, i.e., t_link_{ij} for all $link_{ij}$ in L,
3. the set of all SFs and non-order-constrained SFs, i.e., F and F^*,
4. the locations and the measured processing latencies of all SFs, i.e., $l(f_{ij})$ and t_f_{ij} for all f_{ij},
5. a commonly-used SFP, i.e., $(rf_1, rf_2, \ldots, rf_k)$, $rf_{i=1\sim k} \in F$,
6. a starting SFF s_i, $s_i \in S$,

the goal is to

1. determine the weighted multipath to distribute the traffic load of SFC requests that start at SFF s_i and have SFP $(rf_1, rf_2, \ldots, rf_k)$, so as to minimize the weighted average path latency for these SFC requests. Supposing that the latency of the i-th path $path_{ni}$ for SFC r_n, is denoted by t_path_{ni}, and the weight (probability of selection) of $path_{ni}$ is denoted by w_{ni}, then the weighted average latency of k paths, $avg\_t\_path_n$, can be expressed as

$$avg\_t\_path_n = \sum_{i=1}^{k}(w_{ni} \times t\_path_{ni}) \;/\; \sum_{i=1}^{k} w_{ni}.$$

2. transform the weighted multipath to forwarding entries to be proactively installed in SFFs. This objective involves the design of SFC encapsulation protocol and forwarding table of SFF.

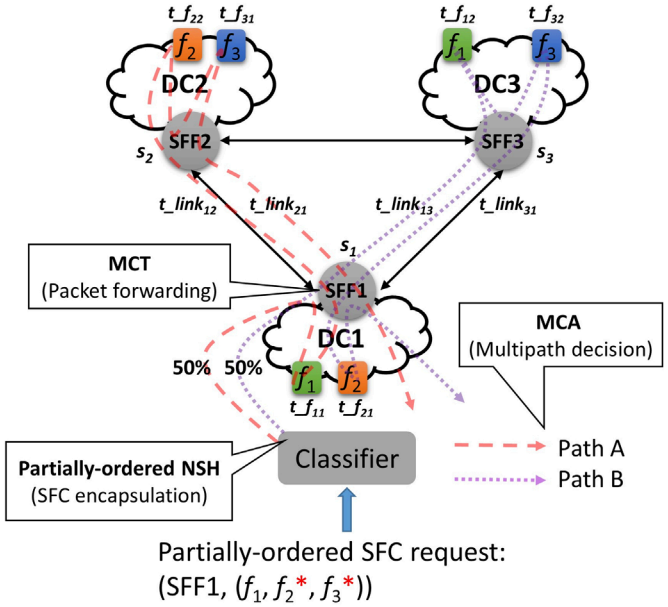


Fig. 2. The architecture of the proposed MCPON.

IV. Solution Design

The proposed MCPON solution is aimed at achieving low-latency service function chaining across multiple datacenters. To eliminate forwarding path decision delays and to provide low end-to-end service latency, MCPON proactively computes and installs multiple forwarding paths for each given commonly-used SFP, by applying a multipath chaining algorithm (MCA), which determines weighted multipath based on a global view of the inter-datacenter network. In practice, the path weights have to be continuously updated according to the network status.

MCPON adopts NSH-Based service chaining. Each SFC request from a user will be encapsulated with an SFP and directed to the SFF of the nearest edge datacenter (by a classifier), for service delivery along one of the predefined multiple paths. Since we assume that some SFs are not order-constrained, the possible forwarding paths of a commonly-used SFP are allowed to have different execution orders for cascading non-order-constrained SFs, so as to increase path diversity and to achieve better load balancing. Thus, SFC requests that have the same SFP and the same starting SFF will be matched by the same forwarding entries but may be forwarded along different paths, and the actual execution order of SFs depends on the path to be traversed.

Fig. 2 shows the architecture of the proposed MCPON, which involves management plane design (MCA for multipath decision) and data plane design (partially-ordered NSH and MCT for SFC encapsulation forwarding). In Fig. 2, we assume all network links have the same bandwidth, and all SF instances have the same processing capacities. For an SFC $r_1 = (s_1, (f_1, f_2^*, f_3^*))$, we can choose to use two paths $path\_A$ ($[s_1, f_1, s_1, s_2, f_2, s_2, f_3, s_2, s_1]$) and $path\_B$ ($[s_1, s_3, f_1, s_3, f_3, s_3, s_1, f_2, s_1]$) with equal probability to balance the incoming request flows. Note that $path\_A$ and $path\_B$ have different execution orders of f_2^* and f_3^*, and even traverse totally different network links and SF

instances. Suppose network link latency and SF processing latency increase as the traffic load increase. Then compared with single-path chaining (i.e., using only *path_A* or *path_B*), two-path chaining can reduce average path latency after injection of request traffic flows.

However, a partially-ordered SFP (flexible execution order of some SFs in an SFP) will not be supported by the original NSH. To support such a partially-ordered SFP, MCPON devises a partially-ordered NSH by modifying the usage of SPI/SI in the original NSH design, so that the SFC encapsulation supports different execution order of SFs for a SPI.

To forward packets encapsulated with the partially-ordered NSH, we also design a multipath chaining table (MCT) in SFF which consults modified SPI/SI values to determine possible next hops. Furthermore, MCT enables forwarding entries to be shared among different SFC requests, reducing forwarding entry consumption.

Below we elaborate the proposed partially-ordered NSH, and then illustrate the design of the multipath chaining table (MCT). Finally, we explain the multipath chaining algorithm (MCA).

A. Partially-ordered NSH

In SFC architecture, an NSH is inserted by the initial classifier at the start of an SFP, and removed at the end of an SFP by the last SFF. The NSH is composed of a 4-byte Base Header (information about the service header and the payload protocol), a 4-byte Service Path Header (path identification and location within a service path), and optional Context Headers (metadata carried along a service path).

To support partially-ordered SFP, MCPON modifies the usage of the Service Path Header in the original NSH design.

1) Modifications on Service Path Header

The Service Path Header contains a 3-byte Service Path Identifier (SPI) and a 1-byte Service Index (SI). SPI uniquely identifies an SFP, and SI provides the location within the SFP. The initial classifier sets the appropriate SPI (a path ID for a given classification result) and the initial value of SI to 255 (or the length of the given SFP), and sends the packet to the first SFF (in the identified SFP) for forwarding along a service path. SFFs can determine the next SF or SFF in the service path according to the SPI/SI values. The SI will be decremented by 1 by the serving SF after performing the required service. In the original NSH design, the combination of SPI and SI provides the identification of an SF and its order within the service plane.

To support partially-ordered SFP, MCPON adopts the proposed partially-ordered NSH, in which the SPI and SI are respectively replaced by a set of SF descriptions and a mask. Each bit in SI is mapped to an SF in the SPI and records the execution status of the corresponding SF. For execution status, 1 signifies executed, and 0 unexecuted. We assign each bit in SI a position number ranging from 0 (for the most significant bit) to 7 (for the least significant bit). Fig. 3 illustrates the updates to the SI mask by SFs. For a classifier to encapsulate an SFP of length 5 $(f_2, f_7, f_3^*, f_5^*, f_6)$, five of the most significant bits (position number 0 to 4) of the SI are mapped to the five

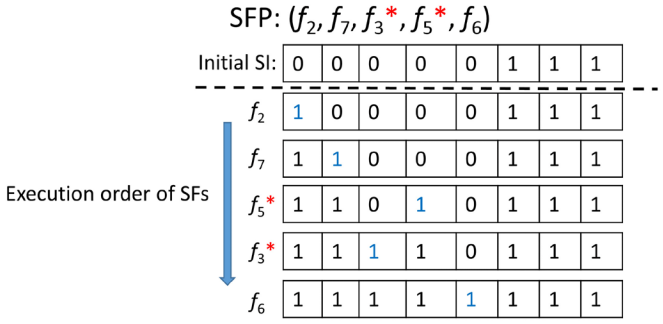


Fig. 3. The updates to SI mask by SFs.

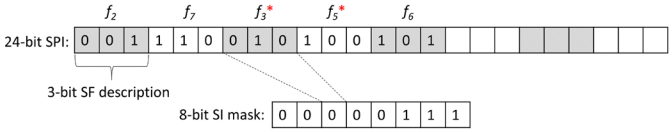


Fig. 4. The mapping of a bit in SI to an SF description in SPI by default.

SFs and are initially set to 0, and the remaining three least significant bits (position numbers 5 to 7) are set to 1. Each zero bit in SI will be set to 1 by the corresponding SF after service execution, as shown in Fig. 3.

To map each bit in the 8-bit SI to an SF description in the 24-bit SPI, we assign 3 bits to each SF description by default, which means the system can support 8 SF types in total. The mapping of a bit in SI to an SF description in SPI is shown in Fig. 4. For updating the SI mask, an SF can obtain a position number by searching its assigned description in the SPI with a sliding window of size 3, which moves right by three positions each time from the most significant bits to the least significant bits. Then the SF will set value 1 to the bit located at the obtained position in SI after service execution. For example, in Figs. 3 and 4, SF f_3^* can obtain position 2 after searching for its assigned description 010 in the SPI, so as to set value 1 to the bit located at position 2 in SI after service execution.

In partially-ordered NSH, the combination of SPI and SI provides an unexecuted-SF list, which is consulted by the proposed MCT in SFF in order to determine the next SF or SFF.

The modification on NSH may affect the packet matching efficiency of an SFF, because an SFF needs to compute the unexecuted-SF list according to the modified NSH for MCT matching. But this issue involves hardware logic design and is beyond the scope of this paper.

2) Reclassification for scalability with Context Header

By default, the system supports 8 SF types in total because of the 3-bit SF description. To support more than 8 SF types in the system, we can assign more than 3 bits to each SF description. However, expansion of SF description will result in a shrinkage of the number of SFs that an SPI can accommodate. For example, to support 64 SF types in the system, we need a 6-bit SF description, and there will be at most 4 (24 divided by 6) SF descriptions in an SPI. To serve an SFC request which has SFP length larger than what an SPI can

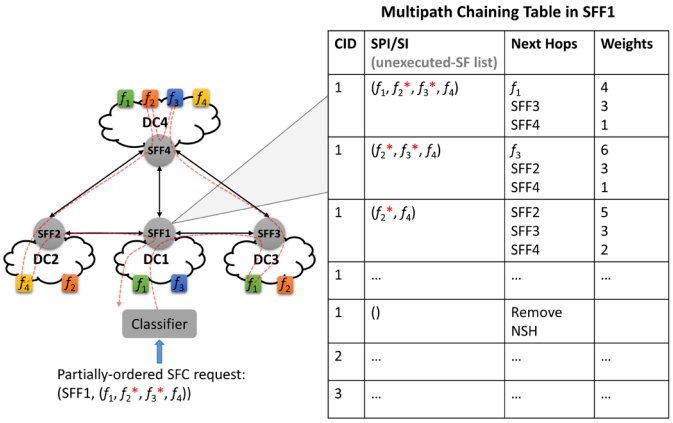
SFP	(SFF$_1$, (f_1, f_2*, f_3*, f_C, f_4, f_5, f_6*, f_C, f_7*, f_8, f_9))
SPI	f_1, f_2*, f_3*, f_C
SI	00001111
Metadata	f_4, f_5, f_6*, f_C, f_7*, f_8, f_9

(a) The Initial NSH

SFP	(SFF$_1$, (f_4, f_5, f_6*, f_C, f_7*, f_8, f_9))
SPI	f_4, f_5, f_6*, f_C
SI	00001111
Metadata	f_7*, f_8, f_9

(b) The updated NSH after 1st reclassification

Fig. 5. The replacement of NSH caused by reclassification in a system with a 6-bit SF description.



CID	SPI/SI (unexecuted-SF list)	Next Hops	Weights
1	(f_1, f_2*, f_3*, f_4)	f_1 SFF3 SFF4	4 3 1
1	(f_2*, f_3*, f_4)	f_3 SFF2 SFF4	6 3 1
1	(f_2*, f_4)	SFF2 SFF3 SFF4	5 3 2
1	...	...	...
1	()	Remove NSH	
2	...	...	...
3	...	...	...

Fig. 6. The layout of the multipath chaining table (MCT).

support, we make use of the reclassification feature for scalability with metadata design in variable-length Context Header.

The SFC architecture [1] supports reclassification as well, typically performed by a classification function co-resident with an SF (or a reclassification SF). As packets traverse an SFP, reclassification may occur, which results in a change of SFP (a replacement of SPI/SI) or an update of the associated metadata. The metadata in the Context Headers provide the ability to exchange context information between classifiers and SFs, and among SFs.

In our design, if the length of a given SFP exceeds the number of SFs that an SPI can accommodate, we first cut the SFP into segments and insert a reclassification SF between every two connected SFP segments to generate a new SFP. We assume that if an SPI can accommodate *n* SFs, the SFP segment length would be equal to *n* - 1. The initial classifier then puts the first *n* SFs in the new SFP to SPI and puts the remaining SFs (in the new SFP) to metadata. Further, *n* of the least significant bits of the SI are set to 0, and the remaining bits are set to 1. As for each reclassification SF, it always retrieves a new SFP from the metadata of the received NSH packet and updates the NSH likewise based on the new SFP.

Fig. 5 illustrates the replacement of NSH resulting from reclassification in a system with a 6-bit SF description. As noted above, an SPI can accommodate 4 SFs with a 6-bit SF description, with the SFP segment length equal to 3. For a given SFP of length 9 (f_1, f_2*, f_3*, f_4, f_5, f_6*, f_7*, f_8, f_9), it will be divided into 3 segments, and 2 reclassification SF f_c will be inserted between SFP segments. Since the new SFP will be (f_1, f_2*, f_3*, f_c, f_4, f_5, f_6*, f_c, f_7*, f_8, f_9), the initial classifier then allocates the first 4 SFs (i.e., f_1, f_2*, f_3*, f_c) to SPI and the remainder of SFs (i.e., f_4, f_5, f_6*, f_c, f_7*, f_8, f_9) to metadata, and finally sends the packet to an SFF for forwarding, as shown in Fig. 5(a). Once the first reclassification SF f_c receives NSH packets (which means f_1, f_2*, f_3* has been executed), f_c will fetch the new SFP from the metadata (i.e., f_4, f_5, f_6*, f_c, f_7*, f_8, f_9), and then likewise update the NSH, as shown in Fig. 5(b), by allocating the next 4 SFs (i.e., f_4, f_5, f_6*, f_c) to SPI and the rest of SFs (i.e., f_7*, f_8, f_9) to metadata, and also resetting the SI.

3) *Modifications on Base Header*

As set out above, the Base Header provides information about the service header. A Version field can be used to indicate whether an NSH is a partially-ordered version or not. To carry variable-length metadata along a service path for reclassification, the Metadata (MD) Type should be set to 0x2, which means zero or more Variable-Length Context Headers may be added immediately following the Service Path Header. We also use the 5 unassigned bits to identify the datacenter ID (CID), which will be discussed below.

B. *Multipath Chaining Table (MCT)*

Fig. 6 shows the layout of a multipath chaining table (MCT). MCT determines possible next hops (i.e., SF or SFF) for partially-ordered NSH packets, based on an unexecuted-SF list and a starting edge datacenter ID (CID, and can be regarded as the index of SFF). The unexecuted-SF list can be obtained by applying an SI mask to the SPI, and the CID is encapsulated in the Base Header by an initial classifier. The table lookup may return more than one possible next hop - essentially a series of weighted paths to be used for load distribution. With MCT, different SFC requests that have the same unexecuted-SF list (may be indicated by different combinations of SPI and SI) and the same starting SFF (CID) can be served by the same shared forwarding entries but may be forwarded to different paths, so as to reduce the forwarding entry consumption.

Furthermore, once the unexecuted-SF list of an encapsulated packet is empty (i.e., all bits of SI are 1), the packet will be forwarded to the final SFF (which is the same as the starting SFF) based on the CID, and be decapsulated by the final SFF.

C. *Multipath Chaining Algorithm (MCA)*

To provide low end-to-end service latency for each given commonly-used SFP, a multipath chaining algorithm (MCA) takes network latencies and SF processing latencies into consideration, computing weights for multipath routing and determining forwarding entries (in SFFs) for load distribution, based on a global view of the inter-datacenter network. Fig. 7

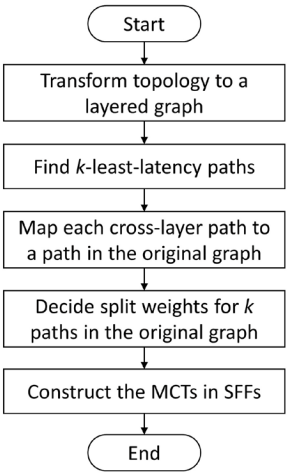


Fig. 7. The flowchart of the multipath chaining algorithm (MCA).

shows the flowchart of MCA. With given parameters as set out in Section III, the MCA first makes use of an ASR algorithm [21] to transform the network graph to a layered graph, so that conventional path algorithms can be used to find paths that traverse all requisite SFs in the correct order. In a network graph transformation, for a given SFP $(rf_1, rf_2, \ldots, rf_k)$, k layers will be added to the original graph (i.e., $k + 1$ layers in total). Each layer is an exact copy of the original graph, and every node (SFF) in the i^{th} layer has a layer qualifier i. The first layer is the base layer of the network topology, G. The $(i + 1)^{th}$ layer ($i \geq 1$) is connected vertically to the i^{th} layer only through SFFs $l(f_{ij})$ that provide rf_i. The cost of each link within a layer is defined by the corresponding network latency (e.g., t_link_{ij}), and the cost of each vertical (cross-layer) edge is defined by the corresponding processing latency (e.g., t_f_{ij}). If there are cascading non-order-constrained SFs in the given SFP, the MCA will find all feasible orders and construct the layered graph as per the specified order or just a layered graph with branches. For more details, the reader is referred to [21].

The MCA then greedily finds k-least-latency paths on the layered graph by Yen's algorithm [28], from the starting SFF s_i in the first layer (source node) to the s_i in the last layer (destination node). Note that the cross-layer paths may not be edge-disjoint (i.e., they may share some edges). After this, the MCA maps each cross-layer path to a path in the original graph, by folding nodes with layer qualifiers back into their base nodes, according to [21]. A feasible forwarding path in the original graph is a list of SFFs and SFs, e.g., $[s_i, f_1, s_i, s_j, f_2, s_j, \ldots, s_i]$. Different paths may have different execution orders of cascading non-order-constrained SFs.

The next step is to determine the weights for all paths in the multipath routing in the original graph. Since we proactively compute and install multiple paths for the given commonly-used SFP without bandwidth demand, MCA only needs to calculate path weights based on current network status, i.e., the path latency derived along with each path by Yen's algorithm. Similar to [29] and [30], the idea of latency-aware traffic load distribution in MCA is to have the path weight inversely proportional to the path latency to achieve load balancing. To be specific, for an SFC request r_n, we denote the least common multiple (LCM) of all path latencies by t_lcm_n, and denote the latency of the i-th path $path_{ni}$ by t_path_{ni}. Then the weight of $path_{ni}$ (denoted by w_{ni}) can be expressed as

$$w_{ni} = t\_lcm_n \,/\, t\_path_{ni}.$$

In practice, to maintain the efficiency of multipath, the path weights should be periodically recalculated according to the measured path latencies.

Finally, MCA constructs the MCTs in SFFs, by transforming the multiple weighted paths to forwarding entries to be installed in several SFFs. As mentioned in Section IV *B*, a packet is matched against the starting CID and the unexecuted-SF list, and will be forwarded to one of the possible next hops (i.e., SF or SFF) by the matching entry, based on weights of the next hops.

Fig. 8 shows the flowchart of MCT construction, and Algorithm 1 presents the pseudocode version of Fig. 8. In Algorithm 1, to generate forwarding entries for the given SFP and starting SFF (CID), MCA sequentially adds or updates forwarding entries for each feasible weighted path. For each path iteration (line 1), the variable, unexecuted-SFP (which is an unexecuted-SF list) is initialized to the given SFP (line 2), and the path will first be transformed into edges (line 3). For example, a forwarding path $[s_1, f_1, s_1, s_2, f_2, s_2, s_1]$ for a request $r_1 = (s_1, (f_1, f_2))$ will be converted into 6 edges $[s_1, f_1]$, $[f_1, s_1]$,

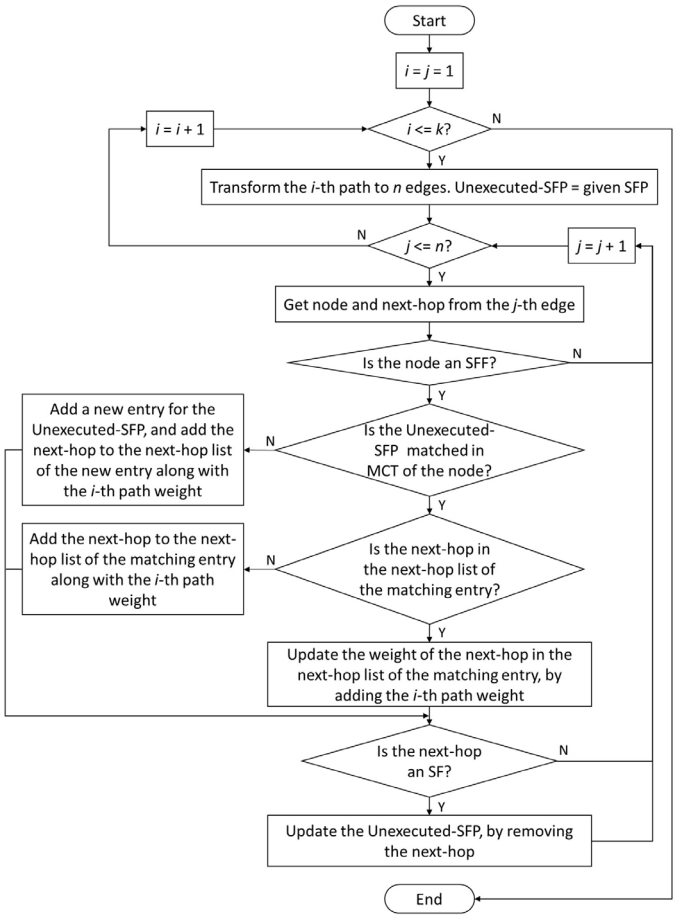


Fig. 8. The flowchart of MCT construction.

Algorithm 1: Generate forwarding entries for the given SFP

```
Input: SFP, weighted_paths
1   for weight, path in weighted_paths:
2     unexecuted_SFP = SFP  # initialization
3     edges = path_to_edges(path)  # transform path to edges
4     for edge in edges:
5       node = edge.src()
6       next_hop = edge.dst()
7       if node is an SFF:
8         if unexecuted_SFP not match in the MCT of the node:
9           # add an entry to the node for matching unexecuted-SFP
10          add_entry(node, unexecuted_SFP, next_hop, weight)
11        else:
12          if next_hop in the output_list of the matched entry:
13            # update the weight of the next-hop in the output_list
14            update_entry_weight(node, unexecuted_SFP,
                                  next_hop, weight)
15          else:
16            # add the next-hop to the output_list with weight
17            update_entry_output(node, unexecuted_SFP,
                                  next_hop, weight)
18        if next_hop is an SF:
19          # update unexecuted-SFP
20          unexecuted_SFP.remove(next_hop)
```

$[s_1, s_2]$, $[s_2, f_2]$, $[f_2, s_2]$ and $[s_2, s_1]$. An edge is considered as a link from a node to its next-hop (e.g., for an edge $[s_1, f_1]$, s_1 is the node and f_1 is the next-hop). After that, MCA iterates through these edges, taking appropriate action for different types of edges.

For each edge iteration (line 4), if the source node of an edge is an SFF (e.g., edges $[s_1, f_1]$, $[s_1, s_2]$, $[s_2, f_2]$ and $[s_2, s_1]$), then MCA first checks whether the CID and unexecuted-SFP match an existing entry in the MCT of the source node (line 8). If not, MCA adds a new entry for the unexecuted-SFP, and adds the next-hop of the edge (an SF or an SFF) to the next-hop list (or output list) of the new entry along with the path weight (line 10); otherwise MCA further checks whether the next-hop exists in the next-hop list of the matched entry (line 12). If not, MCA adds the next-hop to the next-hop list of the matched entry along with the path weight (line 17); otherwise, MCA updates the weight of the next-hop in the next-hop list of the matched entry by adding the path weight (line 14) (i.e., the accumulated weight for the shared edges among paths). Note that a new entry will be added to an MCT only if the CID and unexecuted-SFP do not match any existing entry in the MCT. At the end of edge iteration, MCA checks whether the next-hop is an SF (line 18). If yes (e.g., edges $[s_1, f_1]$ and $[s_2, f_2]$), the unexecuted-SFP will be updated by removing the SF (line 20), and then the process proceeds to the next edge iteration, otherwise it proceeds to the next edge iteration without updating the unexecuted-SFP. After MCA has iterated through all feasible weighted paths, the forwarding entries for the given SFP and starting SFF (CID) are generated.

Since MCT matches the partially-ordered NSH against the starting CID and the unexecuted-SF list, a forwarding entry of an SFP may be reused by some shorter SFPs started with the same CID. Consequently, the forwarding entry consumption can be reduced significantly.

V. Evaluation and Discussion

In our evaluation, we build a full-mesh inter-datacenter network topology with 20 SFFs in a simulation environment (which is the same as the simulation topology used by QGSO [20]). Each SFF is connected directly to each of the others. We assume that each SFF has a local NFV-enabled datacenter, and we randomly place instances of 8 kinds of SFs (f_1 to f_8) to several SFFs. The probability of placing a kind of SF to an SFF is 50%, and it is ensured that every kind of SF (f_i) has at least one instance (f_{ij}) allocated in our topology. The network bandwidth of each link is 10 Gbps, and the processing capacity of each SF instance is also 10 Gbps. In our simulation, we assume that packet arrival to a network link or to an SF follows a Poisson process, and the service time (for both communication and processing) is exponentially distributed. We also assume that the propagation delay is negligible in respect to the queueing delay of the link. Then we can use M/M/1 queuing model [31] to derive latency for both communication and processing (i.e., t_link_{ij} and t_f_{ij}), which is given by:

$$t\_link_{ij} = 1 \,/\, (network\ bandwidth - load),$$
$$t\_f_{ij} = 1 \,/\, (processing\ capacity - load).$$

We set a background traffic load with uniform distribution in the range of 100 Mbps to 1 Gbps to each link and each SF instance.

To evaluate MCPON, 7 commonly-used SFPs of length 2 to 8 are prepared:

1. SFC 1: $(f_1, f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7, f_8)$
2. SFC 2: $(f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7, f_8)$
3. SFC 3: $(f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7)$
4. SFC 4: $(f_1, f_2^*, f_3^*, f_4, f_5)$
5. SFC 5: (f_1, f_2^*, f_3^*, f_4)
6. SFC 6: (f_2^*, f_3^*, f_4)
7. SFC 7: (f_3, f_4)

Note that f_1, f_4, f_7, f_8 are order-constrained, and $f_2^*, f_3^*, f_5^*, f_6^*$ are not order-constrained (the execution order of cascading non-order-constrained SFs could be interchanged). A starting (and ending) SFF is also selected randomly. MCPON then proactively computes forwarding paths and constructs MCTs for the 7 SFCs (the given 7 SFPs starting at the selected SFF), to be evaluated in terms of the saved path computation time, the

TABLE IV
Experiment Configurations

Configuration	Value
Network topology and scale	Full-mesh network, 20 SFFs
Network bandwidth of each link	10 Gbps
The number of types of SFs	8
Processing capacity of each SF instance	10 Gbps
Background traffic load on each link and each SF instance	Uniform distribution in the range of 100 Mbps to 1 Gbps
SFC requests	7 SFPs of length 2 to 8

latency performance and the consumption of forwarding entries. Table IV gives a summary of default experiment configurations.

A. Proactive k-path computation for an SFC of length l at scale of n SFFs saves time complexity of $O(kl^3n^3)$.

We assign 8G RAM, 4 CPUs to the MCA program, and measure the reactive path computation time (which can be saved by proactive path computation), at different scales of inter-datacenter network. Fig. 9 shows the average k-path computation time of the 7 SFCs (k = 1 to 5), at different full-mesh network scales from 20 SFFs (datacenters) to 70 SFFs. The path computation time grows polynomially as a network scale increases. For k = 5, it takes about 8 seconds per SFC at a scale of 70 SFFs.

Recall that MCA finds k-least-latency paths on the layered graph for each SFC by applying Yen's algorithm, which requires $O(kN^3)$ operations. Note that N is the number of nodes in the layered graph. If we denote the SFC length and the number of SFFs in the original network topology by l and n, then N equals $(l + 1)*n$. In other words, by proactive k-path computation, the proposed MCA avoids the on-demand execution of Yen's algorithm, thus saves the time complexity of $O(kl^3n^3)$ for an SFC of length l at scale of n SFFs.

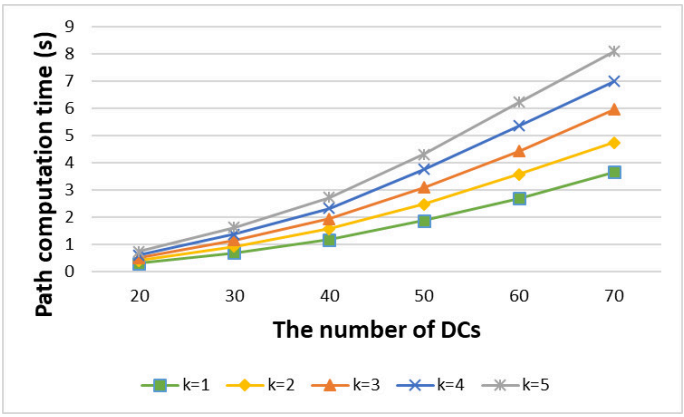


Fig. 9. The average k-path computation time at different network scales.

B. Multipath service chaining reduces latency by 33–68% compared to single-path service chaining.

In this experiment, we measure the latency performance of the 7 SFCs and compare the results with single-path chaining to see how much performance multipath can improve. To evaluate latency performance of the 7 SFCs after MCT construction (at scale of 20 SFFs), we inject each of the 7 SFC flows into the simulation network with random traffic rates in the range of 800 Mbps to 1.2 Gbps (5.6 to 8.4 Gbps SFC traffic is generated in total), and then estimate latency performance of each SFC by computing weighted average latency of k paths of the SFC (i.e., $avg\_t\_path_n$ for an SFC r_n, as noted in Section III).

Fig. 10 shows the latency performance of 7 SFCs when using different k (number of paths), and Fig. 11 shows the latency reductions of 7 SFCs compared with single-path service chaining. In Figs. 10 to 13 the length of SFC 1 to SFC 7 is 8 to 2, respectively. These results indicate that when k = 6, the latencies decreased by 33–68% (46% on the average) compared to k = 1 (single path). On the other hand, the latencies of k = 5 is competitive to k = 6 which suggests that k can be set to 5.

We also investigate the influence of inter-datacenter network topology on the latency performance. We build a partial-mesh topology by randomly removing half of the links in the full-mesh topology (at scale of 20 SFFs), and execute MCPON again to construct MCTs for the 7 SFCs. Fig. 12 and Fig. 13 respectively show the latency performance and the latency reductions of 7 SFCs, under the same traffic injection. The results indicate that the latencies increased by 36% on the average in comparison with the full-mesh topology when k = 6, owing to the decrease in path diversity. In addition, k = 4 is competitive to k = 6 for most of the SFCs (except for SFC 1), because there is a high overlap of links between multiple paths

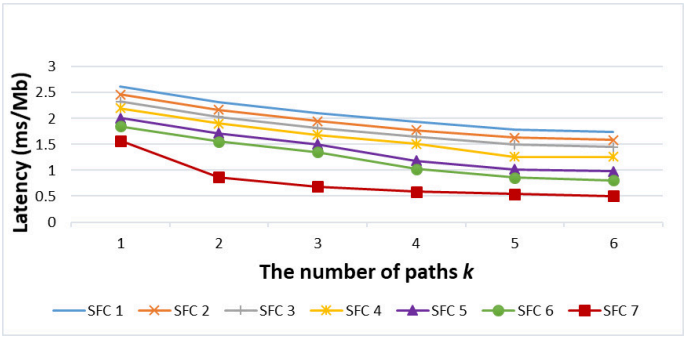


Fig. 10. The latency performance of 7 SFCs when using different number of paths (full-mesh inter-datacenter network).

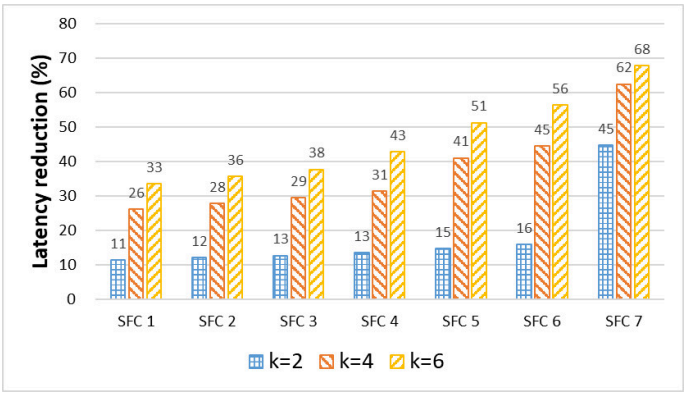


Fig. 11. The latency reductions of 7 SFCs compared with single-path service chaining (full-mesh inter-datacenter network).

when k > 4. In summary, the latencies still decreased by 30–60% (41% on the average) when k = 6 compared to single path.

Fig. 11 and Fig. 13 also show that the ratio of latency reduction increases with the length of SFC in most cases, since there are more feasible paths cloud be selected as k candidate paths for load distribution for longer SFC.

C. MCT reduces forwarding entry consumption by about 36% compared to per-request forwarding.

We also measure the consumption of forwarding entries in MCTs for the 7 SFCs (for the case of full-mesh inter-datacenter network topology), and compare the results with per-request forwarding (no share of forwarding entries between the 7 SFCs) by adjusting the MCT design which matches packets based on the separate SPI and SI values instead of the unexecuted-SF list

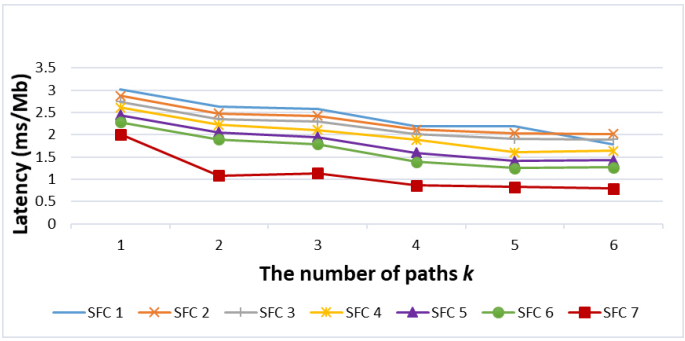


Fig. 12. The latency performance of 7 SFCs when using different number of paths (partial-mesh inter-datacenter network).

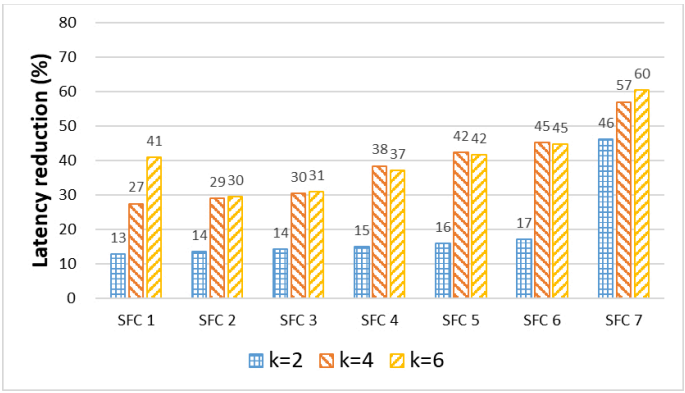


Fig. 13. The latency reductions of 7 SFCs compared with single-path service chaining (partial-mesh inter-datacenter network).

(i.e., every SPI has his own set of forwarding entries). Fig. 14(a) to Fig. 14(c) shows the total system forwarding entry consumption of 7 SFCs when using 1, 3 and 5 paths, respectively. MCPON computes forwarding entries for SFCs from long SFC to short SFC in our experiment, and the accumulated forwarding entry consumption are recorded for each increment of the number of SFCs, as shown in Fig. 14(a) to Fig. 14(c). The horizontal axis in Fig. 14 is the number of SFCs whose forwarding entries are completely installed. The results show that, compared to per-request forwarding, the design of MCT reduces forwarding entry consumption by about 36% on average by sharing forwarding entries among $(f_1, f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7, f_8)$, $(f_2^*, f_3^*, f_4, f_5^*, f_6^*, f_7, f_8)$, and among (f_1, f_2^*, f_3^*, f_4), (f_2^*, f_3^*, f_4), (f_3, f_4). Note that the actual reduction of forwarding entry consumption depends on the similarity between SFCs, i.e., forwarding entries may be shared among SFCs if they start (and terminate) at the same SFF and have the same unexecuted-SF list during service delivery.

D. *Partially-ordered SFP reduces latency by 8-27% compared to fixed-ordered SFP due to the proposed partially-ordered NSH in partial-mesh topology.*

Recall that the proposed partially-ordered NSH enables partially-ordered SFP to increase multipath diversity for better load balancing. In this experiment we investigate the influence of enabling partially-ordered SFP on the latency performance.

To implement fixed-ordered SFP version, we modify the MCA program to use paths that have the same execution order

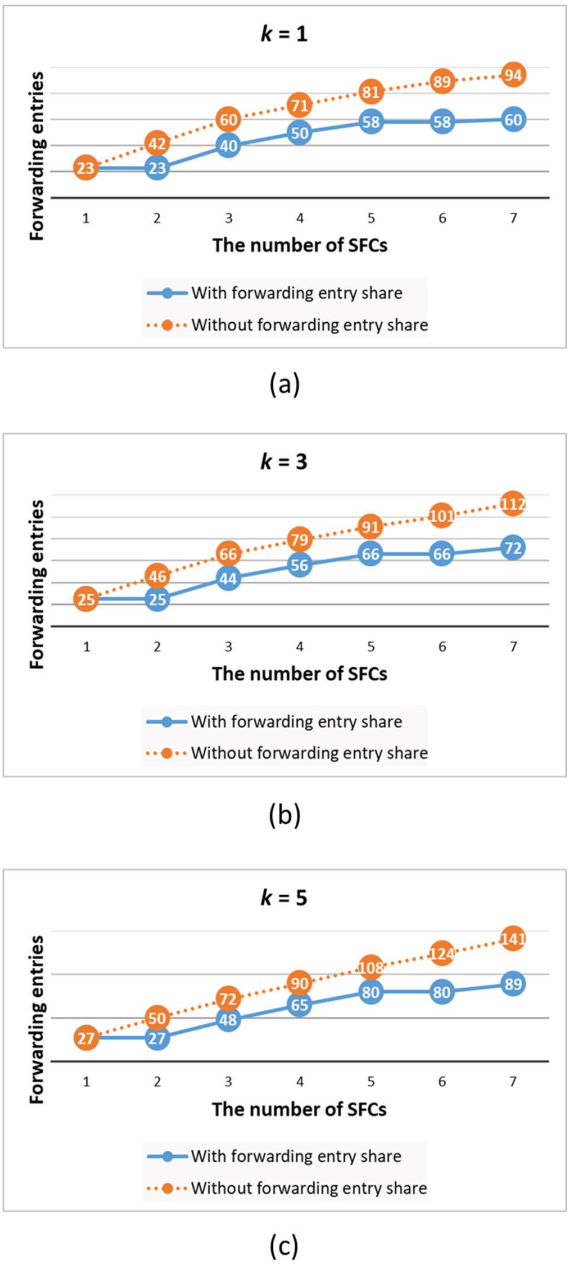


Fig. 14. The forwarding entry consumption of 7 SFCs when using different number of paths.

of SFs as the shortest path, for each SFC.

We execute MCPON (fixed-ordered SFP version) to construct MCTs for the 7 SFCs in full-mesh topology and partial-mesh topology respectively (at scale of 20 SFFs). Note that we build a partial-mesh topology by randomly removing half of the links in the full-mesh topology, and we estimate latency performance of each SFC under the same traffic injection as in our experiment 2. Compared with partially-ordered SFP version (Fig. 10 and Fig. 12), the results indicate that for the fully meshed topology, there is no significant latency difference between using partially-ordered SFP and fixed-ordered SFP, since full-mesh topology still provides enough multipath diversity. As for a general partial-mesh topology, using partially-ordered SFP reduces latencies of SFC

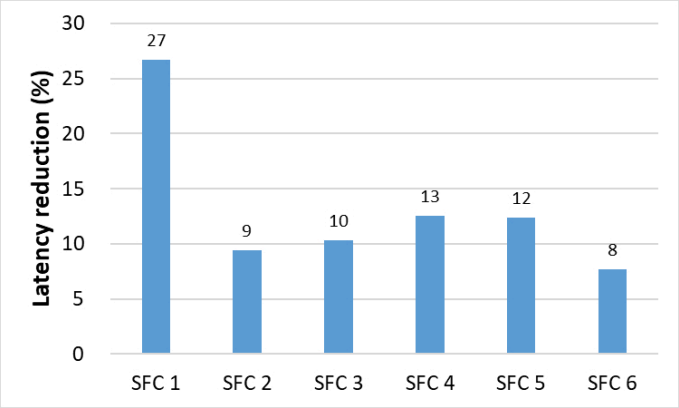


Fig. 15. The latency reductions of 6 SFCs compared with fixed-ordered SFP in partial-mesh inter-datacenter network when using 6 paths.

1 to SFC 6 by about 8-27% (13% on average) compared to using fixed-ordered SFP when $k = 6$, as shown in Fig. 15. In addition, there is no latency difference for SFC 7 because SFC 7 has no cascading non-order-constrained SFs.

VI. Conclusion and Future Work

In this paper we propose an MCPON mechanism, which proactively computes and installs multiple forwarding paths for the commonly-used SFPs, to achieve low-latency service function chaining across inter-datacenter network.

MCPON deploys weighted multipath with the proposed MCA, and adopts a modified NSH which allows the SFC encapsulation to have different execution orders of SFs from the order defined by SPI to support partially-ordered service chaining. We also design an MCT to forward packets which are encapsulated with the proposed partially-ordered NSH. Moreover, MCT reduces the forwarding entry consumption, by enabling forwarding entries to be shared among different SFC requests.

Our evaluations show that proactive k-path computation for an SFC of length l at scale of n SFFs saves time complexity of $O(kl^3n^3)$ as compared to on-demand computation, and multipath service chaining reduces latency by 33–68% compared to single-path service chaining in our simulation scenarios. The evaluation also indicates that MCT reduces forwarding entry consumption by about 36% as compared to per-request forwarding in our simulation environment. Note that the actual reduction of forwarding entry consumption depends on the similarity between SFCs. In addition, the proposed partially-ordered NSH enables partially-ordered SFP to increase multipath diversity for better load balancing, which reduces latency by 8-27% compared to fixed-ordered SFP in partial-mesh inter-datacenter network.

In future work we plan to deploy dynamic path adjustment [32], [33] to adapt to the fast-changing traffic in an inter-datacenter network. Path updating in real-time service delivery is still a key challenge for SFC path selection [27]. Furthermore, if the traffic demands of SFCs are predictable, then proportional routing solutions [34], [35] could also be applied to optimize the network performance. In addition, we also plan to perform a rigorous theoretical analysis on performance of MCPON in our future work.

References

[1] C. Pignataro and J. Halpern, Service Function Chaining (SFC) Architecture, document RFC 7665, Internet Engineering Task Force, Oct. 2015.

[2] P. Quinn and T. Nadeau, “Problem Statement for Service Function Chaining,” IETF RFC 7498, April 2015.

[3] ETSI, “Network Functions Virtualisation – Introductory White Paper,” SDN and OpenFlow World Congress, https://portal.etsi.org/NFV/NFV_White_Paper.pdf, Oct. 2012.

[4] SDN. [Online]. Available: https://www.opennetworking.org/sdn-definition/, Accessed on: March 8, 2020.

[5] M. S. Bonfim, K. L. Dias, and S. F. L. Fernandes, “Integrated NFV/SDN architectures: A systematic literature review,” ACM Comput. Surveys, vol. 51, no. 6, pp. 1–39, Feb. 2019.

[6] Rotsos, Charalampos & Marnerides, Angelos & Magzoub, Abubakr & Jindal, Anish & McCherry, Paul & Bor, Martin & Vidler, John & Hutchison, David. (2020). Ukko: Resilient DRES management for Ancillary Services using 5G service orchestration. 1-6. 10.1109/SmartGridComm47815.2020.9302980.

[7] Jindal, Anish & Aujla, Gagangeet & Kumar, Neeraj & Chaudhary, Rajat & Obaidat, Mohammad & You, Ilsun. (2018). SeDaTiVe: SDN-Enabled Deep Learning Architecture for Network Traffic Control in Vehicular Cyber-Physical Systems. IEEE Network. 32. 66-73. 10.1109/MNET.2018.1800101.

[8] CORD. [Online]. Available: https://www.opennetworking.org/cord/, Accessed on: March 8, 2020.

[9] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar and W. Snow, “Central Office Re-architected as Datacenter,” IEEE Communications Magazine, vol. 54, issue 10, pp. 96–101, Oct. 2016.

[10] Zhang Q, Wang X, Kim I, et al. Service Function Chaining in Multi-Domain Networks[C]//Optical Fiber Communication Conference. Optical Society of America, 2016: Th1A. 6.

[11] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, C. Metz, "COLAP: A Predictive Framework for Service Function Chain Placement in a Multi-cloud Environment", IEEE CCWC, 2017.

[12] P. Quinn, U. Elzur, C. Pignataro, "Network Service Header (NSH)", document RFC 8300, Internet Engineering Task Force, January 2018.

[13] OpenFlow. [Online]. Available: http://www.opennetworking.org/wp-content/uploads/2013/05/TR-535_ONF_SDN_Evolution.pdf/, Accessed on: March 8, 2020.

[14] OpenFlow Switch Specification. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf/, Accessed on: March 8, 2020.

[15] H. Jeong, S. M. Raza, D. Tien Nguyen, S. Kim, M. Kim and H. Choo, "Control Plane Design for Failure Protection in Software Defined Service Function Chains," 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), Taichung, Taiwan, 2020, pp. 1-6.

[16] K. Kannan and S. Banerjee, “Compact tcam: Flow entry compaction in tcam for power aware sdn,” in Distributed Computing and Networking, pp. 439–444, Springer, 2013.

[17] Wang, C.-C.; Lin, Y.-D.; Wu, J.-J.; Lin, P.-C.; Hwang, R.-H. Towards Optimal Resource Allocation of Virtualized Network Functions for Hierarchical Datacenters. IEEE Trans. Netw. Serv. Manag. 2018, 15, 1532–1544.

[18] Y. Chen and J. Wu, “NFV middlebox placement with balanced set-up cost and bandwidth consumption,” in Proc. of ICPP 2018.

[19] C. Pham, N. H. Tran, S. Ren, W. Saad and C. S. Hong, "Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach," in IEEE Transactions on Services Computing, vol. 13, no. 1, pp. 172-185, 1 Jan.-Feb. 2020.

[20] H. Chen et al., ‘‘Towards optimal outsourcing of service function chain across multiple clouds,’’ in Proc. IEEE Int. Conf. Commun. (ICC), May 2016, pp. 1–7.

[21] A. Dwaraki, T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks", Workshop on Hot topics in Middleboxes and Network Function Virtualization (HotMIddlebox), pp. 32-37, 2016.

[22] G. Sun, R. Zhou, J. Sun, H. Yu and A. V. Vasilakos, "Energy-Efficient Provisioning for Service Function Chains to Support Delay-Sensitive Applications in Network Function Virtualization," in IEEE Internet of Things Journal, February 3, 2020.
[23] Y. Ren et al. 2018. On Scalable Service Function Chaining with O(1) Flowtable Entries. In IEEE INFOCOM. IEEE, 702–710.
[24] Dominicini, C.K., Vassoler, G.L., Valentim, R., Villaça, R., Ribeiro, M., Martinello, M., & Zambon, E. (2019). KeySFC: Agile Traffic Steering using Strict Source Routing. Proceedings of the 2019 ACM Symposium on SDN Research.
[25] X. Fei, F. Liu, H. Jin and B. Li, "FlexNFV: Flexible Network Service Chaining with Dynamic Scaling," in IEEE Network, February 11, 2020.
[26] S. Woo et al., "Elastic Scaling of Stateful Network Functions," Proc. 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), Renton, WA: USENIX Association, 2018, pp. 299–312.
[27] H. Hantouti, N. Benamar and T. Taleb, "Service Function Chaining in 5G and Beyond Networks: Challenges and Open Research Issues," in IEEE Network, February 19, 2020.
[28] Jin Y. Yen, "Finding the K shortest loopless paths in a network", Management Science, Vol. 17, No. 11, Theory Series (Jul., 1971), pp. 712-716.
[29] S. Fang, Y. Yu, C.H. Foh, K.M.M. Aung, A loss-free multipathing solution for data center network using software-defined networking approach, in: APMRC, 2012 Digest, IEEE, 2012, pp. 1–8.
[30] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In Proc. ACM SIGCOMM, 2013.
[31] Kleinrock, Leonard, Queueing Systems, Volume I: Theory, New York, Wiley, 1975.
[32] Y.-C. Wang, Y.-D. Lin, and G.-Y. Chang, "SDN-based dynamic multipath forwarding for inter-data center networking," International Journal of Communication Systems, Oct 25, 2018.
[33] Mostafaei, Habib & Shojafar, Mohammad & Conti, Mauro. (2020). TEL: Low-Latency Failover Traffic Engineering in Data Plane.
[34] J. Zhang, K. Xi, L. Zhang, and H. Chao, "Optimizing network performance using weighted multipath routing," in Computer Communications and Networks (ICCCN), 2012 21st International Conference on, pp. 1–7, 30 2012-aug. 2 2012.
[35] S. Nelakuditi, Z. Zhang, D. Du, "On selection of candidate paths for proportional routing", Computer Netw., vol. 44, pp. 79-102, 2004.

Yao-Chun Wang received his Ph.D. degree in computer science from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2021. He is currently a Researcher with Chunghwa Telecom Laboratories (CHTTL), Taoyuan, Taiwan.

Ren-Hung Hwang received his Ph.D. degree in computer science from the University of Massachusetts, Amherst, Massachusetts, USA, in 1993. During 1993~2022, he joined the Department of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan, where he was a distinguished professor of the Department of Computer Science and Information Engineering. He served as the Dean of the College of Engineering during 2014~2017 and the Dean of the Information Technology Office during 2020~2022. In 2022, he joined the College of Artificial Intelligence, National Yang Ming Chiao Tung University, where he is now a professor and the Dean of the AI College. He is also jointly appointed with Research Center for Information Technology Innovation, Academia Sinica. He is currently on the editorial boards of IEEE Communications Surveys and Tutorials and IEICE Transactions on Communications. Prof. Hwang published more than 270 international journal and conference papers. He received the IEEE Best Paper Award from the International Conference on Internet of Vehicles 2019, IEEE Ubi-Media 2018, IEEE SC2 2017, IEEE IUCC 2014 and the IEEE Outstanding Paper Award from IEEE IC/ATC/ICA3PP 2012. Recently, he also served as the general chair of several international conferences on computer networks, including IEEE DataCom 2019, IEEE ISPAN 2018, IEEE SC2 2017, and ICS 2016. His research interests include deep learning, network security, wireless communications, Internet of Things, cloud and edge computing.

Ying-Dar Lin is a Chair Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010-2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017-2018. He cofounded L7 Networks Inc. in 2002, later acquired by D-Link Corp. He also founded and directed Network Benchmarking Lab (NBL) from 2002, which reviewed network products with real traffic and automated tools, also an approved test lab of the Open Networking Foundation (ONF), and spun off O'Prueba Inc. in 2018. His research interests include machine learning for network security, wireless communications, network softwarization, and mobile edge computing. His work on multi-hop cellular was the first along this line, and has been cited over 1000 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), ONF Research Associate (2014-2018), and received K. T. Li Breakthrough Award in 2017 and Research Excellence Award in 2017 and 2020. He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST) with impact factor increased from 9.22 to 25.249 during his term in 2017-2020. He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).