A Mechanism for Configurable Network Service Chaining and Its Implementation

Gang Xiong [1*], Yuxiang Hu [1], Julong Lan [1] and Guozhen Cheng [1]
[1] National Digital Switching System Engineering & Technology Research Center, Zhengzhou, Henan, P.R. China
[e-mail: xg1226@126.com]
[e-mail: chxachxa@126.com]
[e-mail: ndscljl@163.com]
[e-mail: guozhencheng@hotmail.com]
*Corresponding author: Gang Xiong

Received February 4, 2016; revised June 14, 2016; accepted July 27, 2016; published August 31, 2016

Abstract

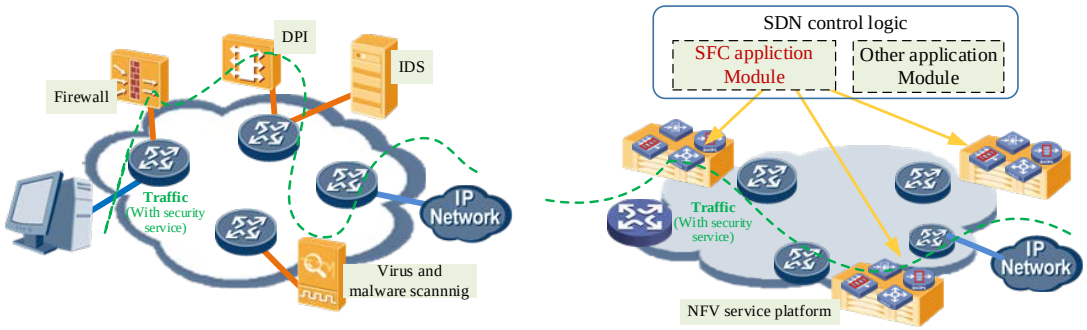
Recently Service Function Chaining (SFC) is promising to innovate the network service mode in modern networks. However, a feasible implementation of SFC is still difficult due to the need to achieve functional equivalence with traditional modes without sacrificing performance or increasing network complexity. In this paper, we present a configurable network service chaining (CNSC) mechanism to provide services for network traffics in a flexible and optimal way. Firstly, we formulate the problem of network service chaining and design an effective service chain construction framework based on integrating software-defined networking (SDN) with network functions virtualization (NFV). Then, we model the service path computation problem as an integer liner optimization problem and propose an algorithm named SPCM to cooperatively combine service function instances with a network utility maximum policy. In the procedure of SPCM, we achieve the service node mapping by defining a service capacity matrix for substrate nodes, and work out the optimal link mapping policies with segment routing. Finally, the simulation results indicate that the average request acceptance ratio and resources utilization ratio can reach above 85% and 75% by our SPCM algorithm, respectively. Upon the prototype system, it is demonstrated that CNSC outperforms other approaches and can provide flexible and scalable network services.

Keywords: Service chain; software-defined networking; network function virtualization; network security

1.Introduction

Nowadays, legacy network services (or functions) are mainly introduced by a wide spectrum of specialized appliances or middleboxes (e.g. Firewalls, Deep Packet Inspection (DPI), Intrusion Prevention/Detection System (IPS/IDS)). However, as shown in **Fig. 1**(a), traditional service models are limited to accommodate diverse requirements, since that these hardware-based and proprietary appliances are costly and difficult to manage in an optimal fashion or to scale up and down with shifting demands [1]. Even worse, the ossified way of deployment, which has network services inserted on the data-forwarding path between communicating peers, makes it impossible to reuse and change the service components.

Recently, two new networking concepts, namely Network Functions Virtualization (NFV) [2] and Software Defined Networking (SDN) [3], have emerged aiming at cost reduction, increase of network scalability and service flexibility. Compared to legacy network, the integration of "SDN+NFV" eases the network function innovation via Service Function Chaining (SFC) [4], which is being used to steer flows through appropriate network functions and enforce network service policies (as shown in **Fig. 1**(b)). Taking network security service for example, each network application may require a certain set of security functions (e.g. virtual Firewalls and IDSes) and these security services can be dynamically chained in a particular sequence in order to satisfy the security requirements of application data flows. Particularly, the mechanism of controlling routing through the specified security service is also called Security Service Chaining (SSC) [5]. It is anticipated that with the advantages of capital expenditures (CAPEX) and operating expenses (OPEX), SFC methodology will inevitably become popular in handling network service functions in the near future.



(a) The legacy network environment (b) The network environment of SDN/NFV

Fig. 1. A typical site of network service model in different networks

However, in addition to packet forwarding, network service can buffer, inject or block certain packets, as well as proxy entire connections. When these services are used in an SFC environment, their actions and properties require careful design and extension. On the other

hand, under limited network resources constraints, such as processing, storage of nodes and link bandwidth, service functions process should be conducted carefully to fully utilize network resources. Therefore, making SFC in networks with virtual service is even a more difficult task and proposes some challenges for network administrators.

In this paper, a Configurable Network Service Chaining (CNSC) mechanism based on "SDN+NFV" is proposed to in order to achieve efficient and coordinated control of service functions over a network. The main contributions of this paper are as follows:

- We formulate the problem of network service chaining and propose a new framework for SFC construction by employing SDN, NFV and segment routing.
- We model service path computation involved in CNSC as an integer linear programming problem. As the problem is a NP-hard, a heuristic algorithm based on network utility maximization is developed to approach the optimal solution.
- We evaluate the efficiency of proposed service path computation method, and validate the usability and feasibility of our CNSC mechanism through a realization prototype.

Roadmap: Section 2 states related works. Section 3 describes the network service chaining problem. Section 4 presents a new framework for constructing network service chains, and discusses some implementation issues in SDN and NFV. Section 5 shows the experiments and evaluation; Finally, Section 6 concludes this paper.

2.Related Work

The integration of "SDN+NFV" eases the network service innovation via outsourcing network functions and constructing dynamic service function chains. The main work related with these research can be summarized as follows:

A. Service Function Chaining Architecture

IETF has taken initiatives towards developing the formal architectures for SFC. The SFC architectural approach proposed by IETF (Quinn and Elzur [6]) suggests implementation of data-plane for supporting network function forwarding. Qazi et al. [7] present a SIMPLE policy enforcement layer based on SDN to efficiently steer middlebox-specific traffic. With OpenFlow protocol [8] (McKeown et al.), Zhang et al. [9] propose a scalable framework (called StEERING) for dynamically routing traffic through any sequence of middleboxes. Fayazbakhsh et al. [10] develop FlowTags architecture to integrate FlowTags-enhanced middleboxes into SDN networks. Further, Gember-Jacobson et al. [11] design a control plane called OpenNF that can provide efficient and coordinated control for reallocation of flows across network functions. Xia et al. [12] address an efficient optical service chaining architecture for network function virtualization in data centers.

B. Service Description and Provision

Service function chain is an abstracted view of a network service that specifies the set of required service functions as well as the order in which they must be executed. First step towards implementation of a service chain is to describe and provide network services. In

aspects of service description, Sun et al. [13] summarize the research on service description languages and enforcement of orchestration policies. Monsanto et al. [14] design a composing language (Pyretic) that can implement network functionality by controlling the flow space of switch in a programmatic manner. In terms of service provision, Shin et al. [15] propose a click-inspired programing framework, called FRESCO, which supports development of modular function programs in an independent SDN controller. Martins et al. [16] provide a ClickOS platform to implement virtual network functions so that network service functions can be migrated from hardware devices to software environment.

C. Service Function Composition Path

A service function composition path (SFCP) is a core mechanism used by service chaining system to express the result of applying more granular policy and operational constraints to the abstract requirements of a service function chain. Baumgartner et al. [17] address the optimization model of mobile core network virtualization. Cheng et al. [18] design a matchmaker supporting composition of higher-level policy modules which operate at a higher layer of the controller stack. Wang et al. [19] develop a combinatorial optimization model to describe the optimization problem of dynamic function composition. Li Y. et al. [21][20] propose a unified service chaining framework that jointly controls and optimizes the resource allocation in SDN/NFV networks. Li T. et al. [21] abstract the service path selection as a grey system theory problem and propose a service composition algorithm to steer network traffics. Hartert et al. [22] provide a declarative and expressive approach to program service functions forwarding in carrier-grade networks.

D. Research Analysis of Service Function Chaining

Various solutions mentioned above so far have addressed the unique and unprecedented challenges imposed by service function chaining. However, these research works are still not perfect and there are some problems for further study as follows:

(a) The architectural approaches still lack analytical models and performance analysis techniques for the proposed solutions. Besides, it is also necessary to emphasize on optimal mapping model of network services to the underlying physical resources, especially considering QoS and SLA constraints.

(b) The composition path methods are confronted with the service configuration complexity [23]. Especially, to optimize network overall performance, it is difficult to solve service function composition problem which usually is a NP hard problem.

Therefore, our work below is enlightened by existing work but is different from them. We make an attempt to address the above challenges in SFC architecture. First we develop a service architecture which can more automatically implement SFC by the instruction of service policies. Then, we theoretically formulate SFC as a constraint programming and provide an optimal mechanism to complete service function chain by a heuristic algorithm.

3.Problem Overview

The objective of this section is to define the problem of applying the service function chaining in SDN/NFV networks.

3.1 Description of Service Function Chaining

Service Function Chaining is a technology or system that efficiently manages the virtual service functions and steers the network traffics through required service functionality. **Fig. 2** shows an SFC architecture where data flows from its employees are forced to traverse the security functions such as LB (Load Balancer), IDS, DPI and firewalls. For example, there are two types of service chains, one for host H1 and H2: LB→IDS→ Firewall, and the other one for host H3: DPI→Firewall.

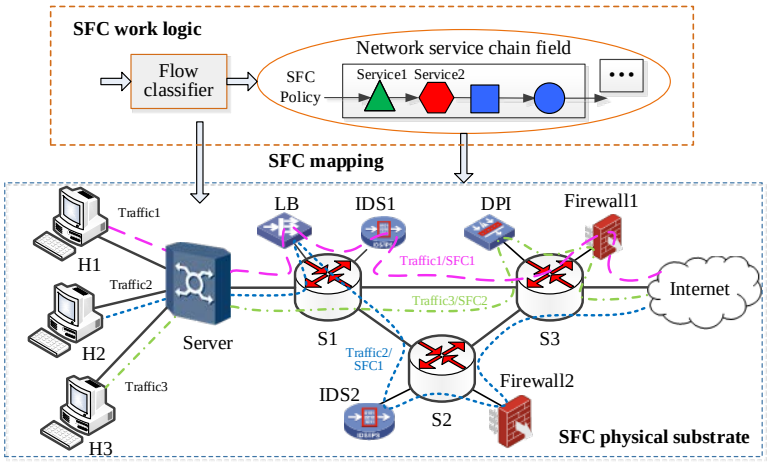


Fig. 2. A architecture of service function chaining

SFC technology possesses some characteristics as follows:

Composability: Due to the standardization of service components and the unified interfaces of various service, SFC can satisfy users' diversity demand by compositing different network service elements, as shown in the SFC work logic of **Fig. 2**. Note that, the composition model may be a linearly sequence chain or a forking one.

Configurability: Each authorized service provider can readily develop and deploy various software-based instances in commodity servers through NFV technology. It is possible that one service has more than one instance in the network, such as the IDS service with two instances "IDS1"and "IDS2" in **Fig. 2**. Therefore, operators can select suitable service instances to configure the service function chain. For example, SFC1 "LB→IDS→Firewall" are implemented by configuring different service instances distributed in the network.

Reusability: A service instance can be shared by multiple service chains. Likely the LB instance in node S1 are used by both traffic1 and traffic2 in **Fig. 2**. In addition, if a new user

wants a set of services which is different from the former one, SFC just needs to reuse some components and adds (or deletes) some ones, and combines them into a new chain for the user.

3.2 Atomic Service and Service Chain Policy

A. Atomic Service Introduction

A service function means a network or application which is used singularly or in concert with other service functions within a service chain to enable a service offered by a network operator. The generic term "L3-L7 services" is often used to describe many service functions. A non-exhaustive list of security service functions includes: Firewalls, IPS/IDS, DPI, server load balancers, network virus and malware scanning, Data Loss Prevention, etc.

A service function is called an Atomic Service (AS) in CNSC. To simplify development of AS, CNSC provides own script language to assist operators in composing ASes from elementary modules. The description language requires the definition of six different variables per instance of modular element: (i) Type, (ii) Input, (iii) Output, (iv) Attributes, (v) Action, and (vi) ID. The Type presents the class of atomic service. The Input/Output denotes the input/output items for an AS. The Attributes are a set of properties for configurations which may contain parameters, performance level, resource cost, etc. The Action represents operation that this AS will perform based on some conditions. The ID is a number generated to identify individual AS instances. Based on the definition of six variables, we can define an AS instance as a six-tuple < *Type*, *Input*, *Output*, *Attributes*, *Action*, *ID*>. For example, **Fig. 3** illustrates a toy example for atomic service instance written in XML format. The AS type is Firewall which monitors the traffic pattern like <dst_port:~80 and dst_IP: 192.168.0.0/16>, and discards all the packets matching the above pattern when detecting malicious profiles.

```
<atomic service>
        <type>Firewall</type>
        <input>All flows </input>
        <output>Normal flows</output>
        <attrs>
            <logic:and>
                <property name="dst_port">
                                ~80
                </property>
                <property name="dst_IP">
                        192.168.0.0/16
                </property>
                <property name="Performance level">
                                High
                </property>
            </logic:and>
        </attrs>
        <action>discard</action>
        <ID>NDSC_Firewall_001</ID>
</atomic service>
```

Fig. 3. The instance of atomic service "Firewall"

B. Service Chain Policy

Generally, a Service Chain Policy (SCP) defines the required atomic services and associated order ($AS_i \rightarrow AS_{i+1}$) that must be applied to packets and/or frames, as shown in the

work logic layer of **Fig. 2**. A service chain policy does not specify the network location or specific instance of atomic services (e.g. firewall 1 vs. firewall 2). We gives the following definition for a service chain policy.

Definition 1 *Service Chain Policy*: $\mathcal{S}=\left\{A_k \middle| k=1,2,\ldots,K\right\}$, $\mathcal{D}=\left\{Demand_j \middle| j=1,2,\ldots,J\right\}$, and $\mathcal{T}=\left\{<source_t, destination_t, business_t> \middle| t=1,2,\ldots,T\right\}$, ($K,J,T\in\mathcal{N}^+$) represent atomic services set, demands set and network traffics set, respectively. A service chain policy is represented as follows:

$$SCP \triangleq <T,S,D>, \left(T\in\mathcal{T}, S\in\mathcal{S}, D\in\mathcal{D}\right) \tag{1}$$

where atomic services set $\mathcal{S}$ is a list of AS identified by function types and each service has several AS instances. Network traffics set $\mathcal{T}$ contains different traffic delivery patterns, where traffic *t* is defined by the source address ($source_t$), destination address ($destination_t$), and business class ($class_t$). Demands set $\mathcal{D}$ defines specific performance profiles, such as the performance levels of ASes.

Each SCP varies with the difference of traffic patterns and the required services. Therefore, a Chain IDentifier (CID) is assigned to mark the service chain police, and the service chain classifier can use CID to determine SCP classification and the service path of traffic through the physical network.

3.3 Instantiation of Service Function Chaining

The instantiation of SFC focuses on mapping between service chain policy and physical network in an optimal way, as shown in **Fig. 2**. The main process involves with the atomic service instances selecting and traffic steering among network nodes respectively. In essence, the instantiation of SFC can be abstracted as the Service Path Computation (SPC) problem [24], where this process should be conducted carefully to fully utilize the network resource. For example, we should distribute flows among different service instances and links for load balance. Selecting service path optimally on demand is more difficult than that in the current networks with the shortest path strategy where only node resource is considered. This complexity is caused by that it is time-consuming for searching optimal service nodes which can provide the instance of each required atomic services, and it is also difficultly in selecting suitable links due to the changes of occupied resources.

Our primary objective is to find out how service chain policies should be employed so that the network utility is maximized. Hence, based on the optimization mathematical model, we formulate service path selection problem as a centralized joint network Resources Utilization Maximization (RUM) problem with some constraints, as follows:

$$\mathcal{P}(SPC) \quad \max_{f\in\mathcal{F}} RUM_f$$
$$s.t. \quad \begin{cases} \forall n \in N^S : \text{Roccupied}(n) \le \text{Rcapacity}(n), \\ \forall l \in L^S : \text{Roccupied}(l) \le \text{Rcapacity}(l), \\ other\ constraint\ conditions. \end{cases} \tag{2}$$

where f is a feasible configuration of network resources and $\mathcal{F}$ is the set of all feasible configuration. N^S and L^S are respectively the node set and link set. $\text{Roccupied}(*)$ denotes the occupied resources on any node or link, and $\text{Rcapacity}(*)$ denotes the resources capacity of any node or link. In what follows, we formulate the service path selection problem as a graph problem, and propose a heuristic algorithm that achieves a proper service instance selection and traffic path steering.

4. Solution Formulation

In this section, we present CNSC mechanism to response the requests of network service chain police.

4.1 System Architecture of CNSC

Based on the main ideas of SDN, the basic idea of our framework is decoupling network control from forwarding hardware, which promises to simplify network management and enables service innovation through network programmability and service function orchestration. In order to implement CNSC in a flexible and efficient way, an administration layer is designed to configure and reconfigure service chain policies. Furthermore, because of the topology-independent of function virtualization technology, we consider a NFV method to provide network services in the data forwarding plane, which can provide service functions with no limit to where they are located. The components of CNSC framework are illustrated in **Fig. 4**. In following content, we detail the role of every component of architecture.

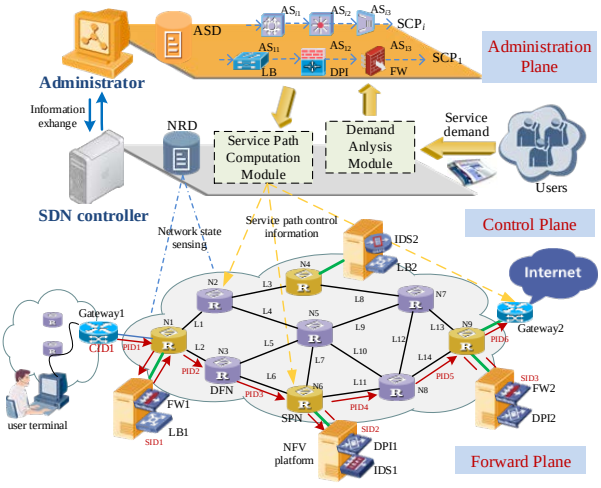


Fig. 4. Framework architecture of CNSC

A. Control Plane

We design the control plane based on SDN controller, such as ONOS [25]. As shown in **Fig. 4**, there are three parts in the control plane. One is the Network Resource Database (NRD) which is responsible for collecting information about network topology and current traffic engineering information and providing information for CNSC computation. Another one is the Service Path Computation Module (SPCM), being responsible for computing the service paths, and the third one is Demand Analysis Module (DAM) which is responsible for accepting service demands from network users (including data centers, enterprises, telecommunication operators, etc.) and provides the analysis results for administration plane to generate service chain policies.

As a core role in our framework, SPCM computes a proper service path for subscriber's requests. Then the proper service function path that chains all of the required service instances is announced to the source nodes (such as gateways). Finally, the source node steers the traffic toward the next node in the service chain by encapsulating the path information into the corresponding packets' header. The detail content will be described in section 4.2.

B.Administration Plane

The administration plane can be viewed as a sub-plane of controller, which aims at simplifying the construction operation of CNSC. There are two roles in administration plane: one is to maintain the registration information of services and provide information inquiry for SCP decision by the Atomic Service Database (ASD). The other one is to configure the specific service chain policies according to service demands.

ASD is used for storing AS information distributed across a network. Through the information exchange between the administrator and the controller, ASD can acquire the information announced by various service providers, which contains instance information of services (e.g. names, function descriptions, versions, sizes, etc.) and location information of service nodes obtained from local routing tables, which means that the service providers turn on the function in service-providing nodes.

The administrator begins to configure the service chain policy after receiving the service request analysis results from the controller. The results include the data traffic state, description of services, or needs for some services in a default order. The administrator inquires ASD for looking up proper services and chooses services that satisfy subscriber's request to achieve the construction of SCP described in section 3.2. **Fig. 4** shows different SCPs in administration plane, one of which is SCP1 for the traffic through Load Balance (LB), DPI, and Firewall. And then these candidate SCPs are delivered to the service path computation module for specifying the service path.

C.Forwarding Plane

The forwarding plane can mainly be divided into three parts: Data-Forwarding Nodes (DFN), Service-Providing Nodes (SPN) and NFV platforms. Data-forwarding nodes are

used for only delivering the traffics among nodes and Service-providing nodes connected with NFV platforms can forward the traffics both among different nodes and between nodes and their corresponding platforms.

NFV platforms provide a virtualization-based method for service implementation. These NFV platforms can be deployed on universal x86 hardware servers instead of a special and expensive device, on which services are embedded through virtual machines at a low cost. As shown in **Fig. 4**, each service runs on one virtual machine, as a result, every service is independent from each other owing to isolation of virtual machines. The administrator just needs to manipulate the virtual machines when he wants to add or delete services.

D. Service Path Expression

As stated above, the path control information from the SPCM is encapsulated into packets' header so that the traffics conforming to the character of CNSC can be steered. In order to reduce the size of path information and improve security, we adopt the Path IDentifier (PID) and Service IDentifier (SID) which are proposed in [21]. A single path with in a domain can be identified by a sole PID consisting of node number and path number. The node numbers can represent different nodes without repeat in a specific domain, and path numbers can denote the links that connect to the nodes. And SID is a service number representing different services implemented on NFV platforms.

As shown in **Fig. 4**, SCP1 is configured with the sequence of services: LB, DPI and Firewall which are named SID1, SID2, and SID3, respectively. According to the service path computation algorithm, the controller finds out a service path for SCP1 which is identified by CID1 and marked as solid red line with arrows. Then, the services composition of CID1 is mapped to path information which is composed of PIDs and SIDs, shown as: {CID1, PID1, SID1, PID2, PID3, SID2, PID4, PID5, SID3, PID6}.

4.2 SPCM Algorithm Modeling

The algorithm used in SPCM enables the joint embedding of individual service chain policy on a substrate network in an optimal way, which can achieve traffic routing optimization (in terms of load balancing, average delay, maximum node resources utility, etc.), while satisfying a correct traversal of service functions for each flow.

In this section, we abstract the service path computation problem as optimization problem and present an SPCM algorithm based on integer linear programming that efficiently finds a routing with a guarantee on the maximum network utility, while satisfying all constraints of network service chain policy. Before modeling SPCM algorithm, we present the related symbols used in this paper and list them for querying conveniently in **Table 1**.

Table 1. Notations of Service Path Computation Problem

Symbol	Annotation
C	One network service chain policy, and $\forall C \triangleq < T^C, S^C, D^C >$
T^C	The network traffic pattern of C and $T^C = \left\langle N_C^{\text{sou}}, N_C^{\text{des}}, B_C \right\rangle$, where N_C^{sou}, N_C^{des}, B_C are the source address, destination address and business class of C, respectively
S^C	The atomic service request set of C and $S^C = \left\{ \left(a_{g-1}^C, a_g^C \right) \middle\| a_g^C \in \mathcal{S}, g = 2,3\ldots, Length\_C \right\}$ where $Length\_C$ is the number of atomic services in C
D^C	The demand indicators set of C and $D^C = \left\{ d_C^{\text{node}}, d_C^{\text{link}} \right\}$ where d_C^{node} (or d_C^{link}) is the node (or link) demand indicators of C
d_C^{node}	$d_C^{\text{node}} \in \left\{ d_C^{NI_k} \middle\| k \in \mathcal{N}^+ \right\}$ where NI_k is the node indicator. We focus on the processing, storage and switching (throughput) capability of network nodes, such that there is $d_C^{\text{node}} = \left\{ d_C^{\text{proc}}, d_C^{\text{stor}}, d_C^{\text{thro}} \right\}$
d_C^{link}	$d_C^{\text{link}} \in \left\{ d_C^{LI_k} \middle\| k \in \mathcal{N}^+ \right\}$ where LI_k is a link indicator. We focus on the link delay, packet loss rate and bandwidth indicators such that there is $d_C^{\text{link}} = \left\{ d^{\text{delay}}, d^{\text{loss}}, d^{\text{band}} \right\}$. Many other indicators can be added in our algorithm easily
G^{S}	The physical substrate network topology and $G^{\text{S}} = (N^{\text{S}}, L^{\text{S}})$
N^{S}	The set of substrate nodes with $n_i \in N^{\text{S}} \left(i = 1,2,\ldots, \left\| N^{\text{S}} \right\| \right)$ ($\|*\|$ is the set cardinality)
L^{S}	The set of substrate links with $l\left(n_i, n_j\right) \in L^{\text{S}} \left(n_i, n_j \in N^{\text{S}}, n_i \neq n_j \right)$
$vol_{n_i}^{NI_k}$	The volume of indicator NI_k on substrate node n_i
$vol_{l(n_i,n_j)}^{LI_k}$	The volume of indicator LI_k on substrate link $l\left(n_i, n_j\right)$
U	The network utility generated by placing demands of network service chain
$u_{n_i}^{\text{N}}$	The network utility generated by substrate node n_i
$u_{l(n_i,n_j)}^{\text{L}}$	The network utility generated by substrate link $l\left(n_i, n_j\right)$
$S(C)$	The set of network service chain policy in network
C_c	The c-th network service chain policy in network and $C_c \in S(C)$
$A(C_c)$	The set of atomic service of C_c and $a_g^c \in A(C_c) \left(g = 1,2,\ldots, Length\_C_c, Length\_C = \left\| A(C_c) \right\| \right)$
$L(C_c)$	The set of service path of C_c and $l\left(a_g^c, a_h^c\right) \in L(C_c) \left(a_g^c \neq a_h^c \right)$
$\delta_{n_i}^{A_r}$	A binary parameter indicating, if node n_i hosts a service of type A_r, $\delta_{n_i}^{A_r} = 1$; else $\delta_{n_i}^{A_r} = 0$
$x_i^{c,g}$	A binary variable indicating, if service a_g^c is mapped into substrate node n_i, $x_i^{c,g} = 1$; else $x_i^{c,g} = 0$
$f_{l(n_i,n_j)}^{l\left(a_g^c, a_h^c\right)}$	A binary variable indicating whether service path $l\left(a_g^c, a_h^c\right)$ passes through substrate link $l\left(n_i, n_j\right)$. $f_{l(n_i,n_j)}^{l\left(a_g^c, a_h^c\right)} \in \{0,1\}$

When the service path computation demands set $S(C)$ are obtained in advance, an integer linear optimization model can be posed whose feasible solution defines a routing that satisfies all network constraints. During service path computation, we select proper substrate nodes with required atomic service instances and substrate links for each chain C_c, such that the total network utility can be improved as much as possible. We formulate model of SPCM problem as follows:

$$\max \quad U(x,f)=\alpha\sum_{n_i\in N^{\mathrm{S}}}\sum_{C_c\in S(C)}\sum_{a_g^c\in A(C_c)}x_i^{c,g}u_{n_i}^{\mathrm{N}}+\beta\sum_{l(n_i,n_j)\in L^{\mathrm{S}}}\sum_{C_c\in S(C)}\sum_{l(a_g^c,a_h^c)\in L(C_c)}f_{l(n_i,n_j)}^{l(a_g^c,a_h^c)}u_{l(n_i,n_j)}^{\mathrm{L}} \tag{3}$$

$$s.t. \quad \sum_{n_i\in N^{\mathrm{S}}}x_i^{c,g}\delta_{n_i}^{a_g^c}=1,\ \forall C_c\in S(C),a_g^c\in A(C_c) \tag{4}$$

$$x_i^{c,g}\le\delta_{n_i}^{a_g^c},\ \forall n_i\in N^{\mathrm{S}},C_c\in S(C),a_g^c\in A(C_c) \tag{5}$$

$$\sum_{C_c\in S(C)}\sum_{a_g^c\in A(C_c)}x_i^{c,g}d_{c,g}^{NI_k}\le vol_{n_i}^{NI_k},\ \forall n_i\in N^{\mathrm{S}},NI_k\in\{\mathrm{proc},\mathrm{stor},\mathrm{thro}\} \tag{6}$$

$$\sum_{C_c\in S(C)}\sum_{l(a_g^c,a_h^c)\in L(C_c)}f_{l(n_i,n_j)}^{l(a_g^c,a_h^c)}d_c^{\mathrm{band}}\le vol_{l(n_i,n_j)}^{\mathrm{band}},\forall l(n_i,n_j)\in L^{\mathrm{S}} \tag{7}$$

$$\sum_{l(a_g^c,a_h^c)\in L(C_c)}f_{l(n_i,n_j)}^{l(a_g^c,a_h^c)}vol_{l(n_i,n_j)}^{LI_k}\le d_c^{LI_k},\forall C_c\in S(C),LI_k\in\{\mathrm{loss},\mathrm{delay}\} \tag{8}$$

$$\sum_{l(n_j,n_i)\in L^S}f_{l(n_j,n_i)}^{l(a_g^c,a_h^c)}-\sum_{l(n_i,n_j)\in L^S}f_{l(n_i,n_j)}^{l(a_g^c,a_h^c)}=x_i^{c,g}-x_i^{c,h},\quad \forall n_i\in N^{\mathrm{S}},C_c\in S(C),l(a_g^c,a_h^c)\in L(C_c) \tag{9}$$

$$x_i^{c,g},f_{l(n_i,n_j)}^{l(a_g^c,a_h^c)}\in\{0,1\},\ \forall n_i\in N^{\mathrm{S}},C_c\in S(C),a_g^c\in A(C_c),l(a_g^c,a_h^c)\in L(C_c) \tag{10}$$

The above Equations (3)-(10) represent the integer linear programming formulation of the service path computation problem, which are consisted in two parts : the objective part described by Equations (3) and the constraints parts described by (4)-(9). We explain them specific meaning as follows:

A. Constraints

(a) *Service uniqueness:* Equations (4) and (5) guarantee that for every network service chain only one service instance of each type is placed, and that the placement is on a unique substrate nodes that are capable to host the respective network service.

(b) *Capacity constraints:* Equation (6) represents the resource volume constraint of the physical nodes. Equation (7) represents the bandwidth resource constraints of the physical links. The QoS (delay and packet loss rate) of service path for each service chain is bounded by Equation (8).

(c) *Flow conservation constraints:* Equation (9) represents that the volume that a flow is sent and received must be one. But for middle nodes, the volume that a flow enters a node equals the volume that a packet exits from it. In addition, Equation (10) is the variables integer constraints.

B. Objective

The main objective of service path computation is to find a set of substrate nodes and

links and design optimal routing approach so as to maximize the resources utility U of substrate network. In this paper, we use the utility function of selected service nodes and links to denote network utility as shown in Equation (3), where α,β (α+β=1) are combination factors. In the selection procedure, the more sufficient the resources volume of network node and link are, the more utility can be generated by them. Thus, node utility $u_{n_i}^{\mathrm{N}}$ and link utility $u_{l(n_i,n_j)}^{\mathrm{L}}$ is defined as:

$$u_{n_i}^{\mathrm{N}} = \exp\left(-\sum_k \omega_k^{\mathrm{N}} \frac{\sum_{C_c \in S(C)} \sum_{a_g^c \in A(C_c)} x_i^{c,g} d_{c,g}^{NI_k}}{vol_{n_i}^{NI_k}} \right), \quad \forall n_i \in N^{\mathrm{S}} \tag{11}$$

$$u_{l(n_i,n_j)}^{\mathrm{L}} = \exp\left(-\left(\sum_k \omega_k^{\mathrm{L}} \frac{\sum_{l \in L(C_c)} f_{l(n_i,n_j)}^{l} vol_{l(n_i,n_j)}^{LI_k}}{d_c^{LI_k}} + \omega_3^{\mathrm{L}} \frac{\sum_{C_c \in S(C)} \sum_{l \in L(C_c)} f_{l(n_i,n_j)}^{l} d_c^{\mathrm{band}}}{vol_{l(n_i,n_j)}^{\mathrm{band}}} \right) \right), \forall l\left(n_i,n_j\right) \in L^{\mathrm{S}} \tag{12}$$

In equations (11) and (12), we use the ratio between demand volume and actual volume on nodes (or links) to express the sufficient degree of each resource indicator. The smaller the ratio value, the more sufficient the resource is. Then, the affine coefficients ω_k^{N} (or ω_k^{L}) are used to combine the different indicator ratio values, where there is $\omega_k^{\mathrm{N}} \geq 0$ and $\sum_k \omega_k^{\mathrm{N}} = 1$ (or ω_k^{L}). Finally, we use an exponential function $\exp(*)$ to transmit the combination ratios so that the utility value $u_{n_i}^{\mathrm{N}}$ (or $u_{l(n_i,n_j)}^{\mathrm{L}}$) increases with the sufficient degree of network resources.

4.3 Optimal Solution of SPCM Algorithm

The optimization problem formulated above belongs to the class of Integer Linear Programs (ILP). In general, this problem is NP-hard, and it is extremely difficult to obtain its solution. Here, to quickly compute good solutions, we propose a *heuristic approach* to construct a feasible service path scheme for SPCM algorithm. This *heuristic* approach framework consists of two parts: node mapping based on service capacity matrix and link mapping with segment routing policies.

A. Node mapping based on service capacity matrix

In the node mapping stage, we design a selection mapping algorithm (*Algorithm* 1) based on node service capacity matrix. It works as follows:

First, we define the service capacity matrix for nodes in network.

Definition 2. *Service Capacity Matrix:* Given a substrate network $G^{\mathrm{S}} = (N^{\mathrm{S}}, L^{\mathrm{S}})$, the service capacity matrix of N^{S} is formulated as $\mathbf{M}^{\mathrm{Nod}} = [m_{i,r}]_{N \times R}$ where $m_{i,r}$ is the service

capacity of node $n_i\left(n_i \in N^{\mathrm{S}}, i=1,2,\ldots,N=\left|N^{\mathrm{S}}\right|\right)$ that can provide for the type of atomic service A_r. Each service capacity $m_{i,r}$ can be calculated by

$$m_{i,r}=u_{n_i}^{\mathrm{N}} \times \delta_{n_i}^{A_r} \tag{13}$$

where $u_{n_i}^{\mathrm{N}}$ is utility value of node n_i and can be obtained by Eq.(11), and $\delta_{n_i}^{A_k}$ is a binary parameter for indicating whether there is service A_r on node n_i.

Second, we construct a service request matrix $\mathbf{M}^{\mathrm{Req}}=[q_{r,r}]_{R\times R}$ for policy C_c. $\mathbf{M}^{\mathrm{Req}}$ is a diagonal matrix, i.e.

$$q_{r,r}=\begin{cases}1, \text{ if } r=r \text{ and } A_r=a_g^c;\\ 0, \text{ else}\end{cases}, \quad a_g^c \in A(C_c)\left(g=1,2,\ldots,\left|A(C_c)\right|\right) \tag{14}$$

Third, we calculate the selection matrix $\mathbf{M}^{\mathrm{sel}}=[e_{i,r}]_{N\times R}$ which contains the required services for policy C_c as:

$$\mathbf{M}^{\mathrm{sel}}=\mathbf{M}^{\mathrm{Nod}} \times \mathbf{M}^{\mathrm{Req}} \tag{15}$$

Each column vector of matrix $\mathbf{M}^{\mathrm{sel}}$ is denoted as $\overrightarrow{e_r}=\left(e_{1,r},\ldots,e_{N,r}\right)^{\mathrm{T}}\ (r=1,2,\ldots,R)$ such that $\mathbf{M}^{\mathrm{sel}}=\left[\overrightarrow{e_1},\ldots,\overrightarrow{e_R}\right]$. If $\overrightarrow{e_r}=0$, service A_r cannot be used for policy C_c; else, node n_i is a candidate node for policy C_c when $\overrightarrow{e_r}\neq 0$ and $e_{i,r}\neq 0\left(e_{i,r}\in\overrightarrow{e_r}\right)$.

Finally, we select the optimal nodes for the policy C_c based on the matrix $\mathbf{M}^{\mathrm{sel}}$. For each $a_g^c \in A(C_c)$ and $A_r=a_g^c$, we choose node n_i as the service node for service a_g^c when the n_i satisfies the condition that is

$$e_{i,r}^{\mathrm{sel}}=\arg\max_i\left\{e_{i,r}\left|e_{i,r}\in\overrightarrow{e_r},\overrightarrow{e_r}\neq 0, n_i \text{ satisfies Eq.}(6)\right.\right\} \tag{16}$$

We use the symbol $n_{i,g}^{\mathrm{sel}}$ to represent the corresponding node of the $e_{i,r}^{\mathrm{sel}}$. All the selected nodes compose the service nodes set Ω^{N}, as follows:

$$\Omega^{\mathrm{N}}=\left\{n_{i,g}^{\mathrm{sel}}\left|g=1,2,\ldots,\right|A(C_c)\right|\right\} \tag{17}$$

As stated as above, we further describe the process of node mapping in **Table 2**

Table 2. Node mapping procedure of SPCM algorithm

Algorithm 1. Node Mapping based on Service Capacity Matrix

1. Compute the service capacity matrix $\mathbf{M}^{\mathrm{Nod}}$ (Eq.(13));
2. Compute the service request matrix $\mathbf{M}^{\mathrm{Req}}$ (Eq.(14));
3. Compute the selection matrix $\mathbf{M}^{\mathrm{sel}}$ (Eq.(15));
4. **for** each a_g^c in sorted $A(C_c)$ **do**
5. $n_g^{\mathrm{sel}}=Null$; // initialization of selected node set
6. **for each** $A_r\ (r=1,\ldots,R)$ **do**;
7. **if** $a_g^c=A_r$ and $\overrightarrow{e_r}\neq 0$ **then do**
8. **While** ($n_g^{\mathrm{sel}}=Null$ and $\overrightarrow{e_r}\neq Null$)
9. $e_{i,r}^{\mathrm{sel}}=\arg\max_i\left\{e_{i,r}\left|e_{i,r}\in\overrightarrow{e_r},\overrightarrow{e_r}\neq 0\right.\right\}$;
10. $n_i \mapsto n_{i,g}^{\mathrm{sel}}$; // node n_i is selected as spare service node of atomic service a_g^c

11. **if** $n_{i,g}^{sel}$ satisfies the constraints Eq.(6)
12. $n_{i,g}^{sel} \mapsto n_g^{sel}$
13. **else**
14. $\overrightarrow{e_r} = \overrightarrow{e_r} \setminus e_{i,r}^{sel}$; // element $e_{i,r}^{sel}$ is deleted from vector $\overrightarrow{e_r}$
15. **end if**
16. **end while**
17. $n_g^{sel} \mapsto \Omega^N$; // node n_g^{sel} is added into set Ω^N
18. **else**
19. $n_g^{sel} \mapsto \Omega^N$
20. **end if**
21. **end for**
22. **end for**
23. **for** each n_g^{sel} in Ω^N
24. **if** $n_g^{sel} = Null$
25. Output mapping failure
26. **end if**
27. **end for**
28. **return** Output mapping success result Ω^N

B. Link Mapping with Segment Routing

In the link mapping stage, we design a link mapping algorithm (*Algorithm* 2) based on Segment Routing (SR) [26] to find the optimal routing scheme. SR provides enhanced packet forwarding capabilities while keeping a low configuration impact on networks. The basic idea of SR is to prepend packets with a stack of labels which is called segments and encapsulated in a segment routing header. A segment including node segment and adjacent segment represents an instruction. There, we focus on node segments that can be used to define paths in a network topology.

Initially, we introduce the definition of segment routing path [22] and use the segment routing path for describing the link mapping of service chain.

Definition 3 *Segment Routing Path* (*SR-path*): Given two nodes $s,t \in N^S (s \neq t)$, a SR-path from s to t is a non-empty sequence of forwarding graphs

$$FG(s,n_1), FG(n_1,n_2), \ldots, FG(n_{i-1},n_i), FG(n_i,t)$$

such that the destination of a forwarding graph is the source of its successor in the sequence. Also, the source of the first forwarding graph and the destination of the last forwarding graph respectively correspond to the source and the destination of the SR-path. A forwarding graph $FG(n_{i-1},n_i)$ describes a flow between a pair of nodes $n_{i-1},n_i \in N^S (n_{i-1} \neq n_i)$ in the network, which is a non-empty directed acyclic graph rooted in n_{i-1} and converged towards n_i. Three SR paths from source node *s* to destination node *t* are illustrated in **Fig. 5**.

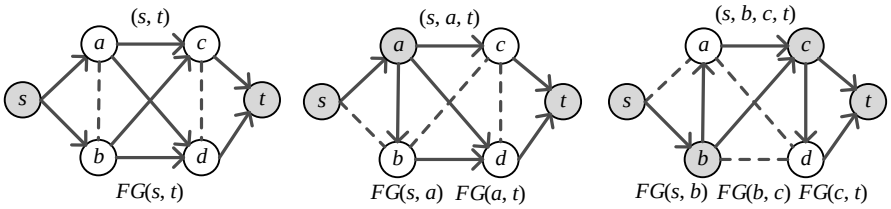


Fig. 5. Three different SR-paths based on the forwarding graphs

In **Fig. 5,** the shadow nodes represent nodes that the segment routing policy requires to pass through. The arrow lines indicate the optional connected paths between the adjacent forwarding nodes. For example **Fig. 5(b)**, the routing policy requires to pass through three nodes (i.e. *s*, *a*, *t*) and the SR path consists of $FG(s,a)$ and $FG(a,t)$. The path set of $FG(a,t)$ includes three optional paths namely $a \to c \to t$, $a \to d \to t$ and $a \to b \to d \to t$.

The aim of the service chaining policy is to force a traffic to traverse a particular sequence of service nodes. Thus, we can use the SR-path to represent the service path. Through the node mapping, the set of nodes providing the corresponding service for policy C_c is obtained from Eq. (17), i.e. $\Omega^{\mathrm{N}} = \left\{ n_{i,g}^{\mathrm{sel}} \middle| g = 1,2,\ldots,\left|A(C_c)\right| \right\}$. Let $N_{C_c}^{\mathrm{sou}}, N_{C_c}^{\mathrm{des}}$ be the source and the destination node of C_c, and the SR-path of the service chaining policy C_c is shown as

$$FG\left(N_{C_c}^{\mathrm{sou}}, n_{i,1}^{\mathrm{sel}}\right), FG\left(n_1, n_2\right), \ldots, FG\left(n_{i,g-1}^{\mathrm{sel}}, n_{i,g}^{\mathrm{sel}}\right), \ldots, FG\left(n_{i,|A(C_c)|}^{\mathrm{sel}}, N_{C_c}^{\mathrm{des}}\right) \tag{18}$$

Then, we calculate the forwarding path for each forwarding graph $FG\left(n_{i,g-1}^{\mathrm{sel}}, n_{i,g}^{\mathrm{sel}}\right)$. According to the substrate network $G^{\mathrm{S}} = (N^{\mathrm{S}}, L^{\mathrm{S}})$, we design a *K* shortest-path algorithm to construct a connected path between $n_{i,g-1}^{\mathrm{sel}}$ and $n_{i,g}^{\mathrm{sel}}$ for each forwarding graph $FG\left(n_{i,g-1}^{\mathrm{sel}}, n_{i,g}^{\mathrm{sel}}\right)$.

First, based on the link bandwidth demand of policy C_c (noted as $band\_demand\left(L(C_c)\right)$), we construct the constraints through Eq.(7) as follows:

$$band\_load\left(l\left(n_i, n_j\right)\right) + band\_demand\left(L(C_c)\right) \le vol_{l\left(n_i, n_j\right)}^{\mathrm{band}}, \forall l\left(n_i, n_j\right) \in L^{\mathrm{S}} \tag{19}$$

where the parameter $band\_load\left(l\left(n_i, n_j\right)\right)$ is the bandwidth occupation of link $l\left(n_i, n_j\right)$. We delete links which do not satisfy the constraint of Eq. (19) from the G^{S} and get the sub-graph $G_1^{\mathrm{S}} = (N^{\mathrm{S}}, L_1^{\mathrm{S}})$ where $\forall l \in L_1^s$ satisfies the constraint of Eq. (19).

Furthermore, based the link utility of Eq.(12), we calculate the utility value for each link by

$$u^{\mathrm{L}}_{l\left(n_i,n_j\right)}=\exp\left(-\left(\omega^{\mathrm{L}}_1\frac{vol^{\mathrm{delay}}_{l\left(n_i,n_j\right)}}{Th^{\mathrm{delay}}_c}+\omega^{\mathrm{L}}_2\frac{vol^{\mathrm{loss}}_{l\left(n_i,n_j\right)}}{Th^{\mathrm{loss}}_c}+\omega^{\mathrm{L}}_3\frac{band\_load\left(l\left(n_i,n_j\right)\right)}{vol^{\mathrm{band}}_{l\left(n_i,n_j\right)}}\right)\right),\forall l\left(n_i,n_j\right)\in L^s_1 \tag{20}$$

where Th^{delay}_c and Th^{loss}_c are the thresholds of delay and loss packet rate required by policy C_c, respectively. Using $-u^{\mathrm{L}}_{l\left(n_i,n_j\right)}$ as link weight value in G^{S}_1, we can obtain the weighted graph G^{w}_1.

And then, we assume that on graph G^{w}_1, a *K* Shortest-Path algorithm (KSP) is used for searching traffic routing between $n^{\mathrm{sel}}_{i,g-1}$ and $n^{\mathrm{sel}}_{i,g}$. KSP is an extended version of the shortest path algorithm. Being different from the shortest path algorithm, KSP can calculate *K* alternative paths between the starting point and end point, and form the shortest path group to meet the user's choice demand.

K shortest-path algorithm can achieve *K* paths which are denoted as $\left\{P_{g\kappa}\middle|\kappa=1,2\ldots,K\right\}$. From the path set, we can select an optimal path P^{FG}_g as the path between $n^{\mathrm{sel}}_{i,g-1}$ and $n^{\mathrm{sel}}_{i,g}$, and P^{FG}_g satisfies

$$P^{\mathrm{FG}}_g=\arg\max_{\kappa}\left\{\sum_{l_{g\kappa}\in P_{g\kappa}}u^{\mathrm{L}}_{l_{g\kappa}}\middle|\sum_{l_{g\kappa}\in P_{g\kappa}}vol^{\mathrm{loss}}_{l_{g\kappa}}\le d^{\mathrm{loss}}_c,\sum_{l_{g\kappa}\in P_{g\kappa}}vol^{\mathrm{delay}}_{l_{g\kappa}}\le d^{\mathrm{delay}}_c\right\},\quad \kappa=1,2\ldots,K \tag{21}$$

Eq. (21) means that $P_{g\kappa}$ where the links has the maximum of utility summation and conform to the conditions of (8) is selected as the path for forwarding graph $FG\left(n^{\mathrm{sel}}_{i,g-1},n^{\mathrm{sel}}_{i,g}\right)$.

Finally, we calculate each path P^{FG}_g ($g=1,2,\ldots,\left|A\left(C_c\right)\right|$) for each forwarding graph and connect all paths as the SR-path of policy C_c as follows:

$$\Omega^{\mathrm{L}}=\left\{p^{\mathrm{FG}}_g\middle|g=1,2,\ldots,\left|A\left(C_c\right)\right|\right\} \tag{22}$$

In summary, we represent the process of link mapping in **Table 3**.

Table 3. Link mapping procedure of SPCM algorithm

Algorithm 2. Link Mapping with Segment Routing

1. Input the selected service node set Ω^{N};
2. Input the network topology $G^{\mathrm{S}}=(N^{\mathrm{S}},L^{\mathrm{S}})$;
3. Represent the SR-path by forwarding graph as Eq. (18);
4. **for each** $FG\left(n^{\mathrm{sel}}_{i,g-1},n^{\mathrm{sel}}_{i,g}\right)$ in SR-path **do**
5. Compute sub-graph $G^{\mathrm{S}}_1=(N^{\mathrm{S}},L^{\mathrm{S}}_1)$ by Eq. (19);
6. Compute weighted graph G^{w}_1 by Eq. (20);
7. Solve path P^{FG}_g for forwarding graph $FG\left(n^{\mathrm{sel}}_{i,g-1},n^{\mathrm{sel}}_{i,g}\right)$ by Eq. (21);
8. **end for**
9. **for** each P^{FG}_g in Ω^{L}
10. **if** $P^{\mathrm{FG}}_g=Null$
11. Output mapping failure
12. **end if**
13. **end for**
14. **return** Output mapping success result Ω^{L} as the SR-path of policy C_c

5.Experiments and Evaluation

In this section, we first evaluate the performance of SPCM algorithm. Then we implement CNSC mechanism in a prototype to illustrate its usability and efficiency.

5.1 Experimental Details

We perform simulation experiments on two topologies to assess the efficiency of SPCM algorithm. The one is a real topology called Internet2 OS3E [27] which is used for advanced scientific researches in USA and contains 34 nodes and 42 links. The other one is a synthetic topology which is generated by GT-ITM tool [28]. The synthetic topology is a power-law random network graph with 100 nodes and the average node degree is set 6. The real topology is meant to assess the efficiency of our approach on practical situations, while the synthetic topology is used to measure the behavior of our approach on complex networks. We implement the proposed algorithms in C++ and execute on a computer equipped with an Intel(R) Core(TM) i7 CPU 2.67GHz processor with 2 cores, and 4GB of RAM.

In the substrate network, we assume each node can be used as the service-providing node. The number of service function classes is set 10 and all kinds of services can be uniformly deployed on each node. For the simulation, the parameters for network resources and each service function are randomly generated with a uniform distribution as follows:

(a) We consider four network resources namely three node resources (processing, storage, throughput) and one link resource (bandwidth). The volume of each network resource with each node (or link) is uniformly distributed between 500 and 1000.

(b) We consider two QoS attributes of each link (i.e. delay and packet loss rate). The number of each attribute uniformly ranges from 1 to 5. For each service instance, the required volume of each network resource are uniformly distributed between 1 and 10.

(c) The service chain requests arrive in a Poisson process. The number of atomic service instances in each service chain is denoted as “Length_C”. It means that each service chain has Length_C different service functions. For each test scenario of “Length_C”, we generate 10 different requests and then report the average value of the all test cases.

5.2 Simulation Results

In the subsection, we measure the efficiency of our approach by analyzing the request acceptance ratio and network resources utilization ratio as follows. And for comparison purposes, we also evaluate the performance of different algorithms.

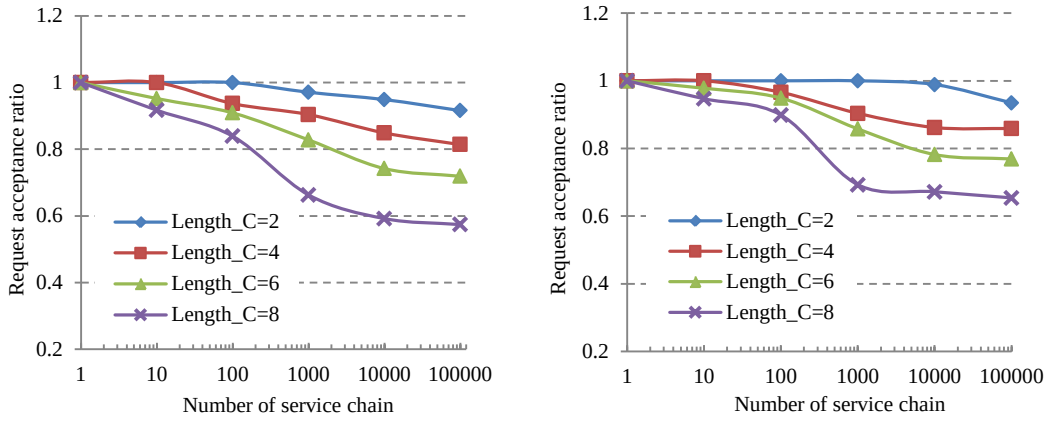
A. Results of request acceptance ratio

The Request Acceptance Ratio (RAR) is defined by the ratio between the number of successfully accepted service chain policies and the number of total service chain request (R), which is calculated by

$$RAR = \lim_{R\to\infty} \frac{\sum_{c=1}^{R} \delta(C_c)}{R} \tag{23}$$

where $\delta(C_c)=1$ if the service chain request C_c is accepted by network, otherwise $\delta(C_c)=0$.

Fig. 6 illustrates the evaluation results of request acceptance ratio with different length of service chain on the two experimental network topologies. Firstly, *RAR* under the same value of Length_C decreases with the increase of the number of service chains in each substrate network. The reason is that the network load is improved when the number of service chains becomes greater. Secondly, the larger Length_C is, the smaller *RAR* is. Because the longer service chains need more atomic service instances which means that more nodes and links are acquired. Finally, comparing **Fig. 6(a)** and **Fig. 6(b)**, we can find that *RAR*s on two topologies have the similar variation trend, which can be attributed to the heuristic search strategy of SPCM algorithm. The solution of SPCM algorithm may be approximate optimal or local optimal and usually complete requests of short service chains in a single node. Moreover, the *RAR* means of the synthetic and real network are respectively 90.4% and 87.8%, so *RAR* of the synthetic topology is generally better than that of the real topology. It is due to that the synthetic topology with more connectivity is helpful to find suitable service path for requests.



(a) RAR results on the real topology (b) RAR results on the synthetic topology

Fig. 6. The results of request acceptance ratio on two topologies

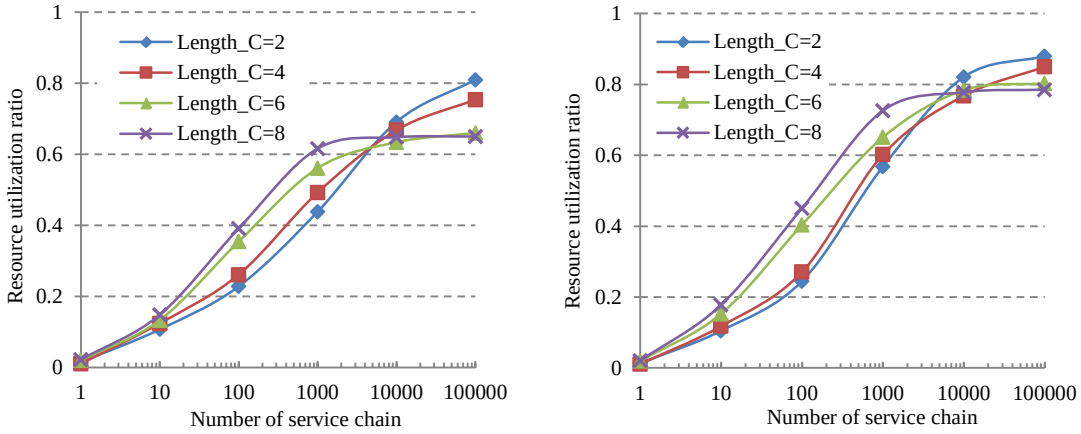
B. Results of resources utilization ratio

The Resources Utilization Ratio (RUR) is defined by the average ratio between the occupied volume of each resource ($vol_{ki}^{\text{occupied}}$) and the total volume of each resource (vol_{ki}^{Total}) on all nodes and links, which is calculated by

$$RUR=\frac{1}{\sum_{i=1}^{(N^S+L^S)}\sum_{k=1}^{K}\delta\left(R_i^k\right)}\left(\sum_{i=1}^{(N^S+L^S)}\sum_{k=1}^{K}\frac{vol_{ki}^{\text{occupied}}\delta\left(R_i^k\right)}{vol_{ki}^{\text{Total}}}\right) \tag{24}$$

where k (k=1,2,…,K) is resource types, N^S and L^S are the total number of nodes and links, respectively. $\delta\left(R_i^k\right)=1$ if node (link) i has k type of resource, otherwise $\delta\left(R_i^k\right)=0$.

Fig. 7 shows the evaluation results of resources utilization ratio on the two experimental network topologies. We observe that RUR increases along with the number of service chains in each topology, and the service chains with smaller length value (Length_C) can obtain higher *RUR*. This is because the short service chains are more easily configured by selecting less nodes and links. Meanwhile, when the number of service chains is 10^5, RURs under different parameters (i.e. Length_C=2, 4, 6, 8) in the real network are 0.81, 0.75, 0.66, 0.65, respectively, while RURs in the synthetic network are 0.88, 0.85, 0.803, 0.785, respectively. By mean calculation, the RUR average values in the real and synthetic network are 71.8% and 82.9%, respectively. Thus, the synthetic network achieves higher resource utilization compared with the real topology.



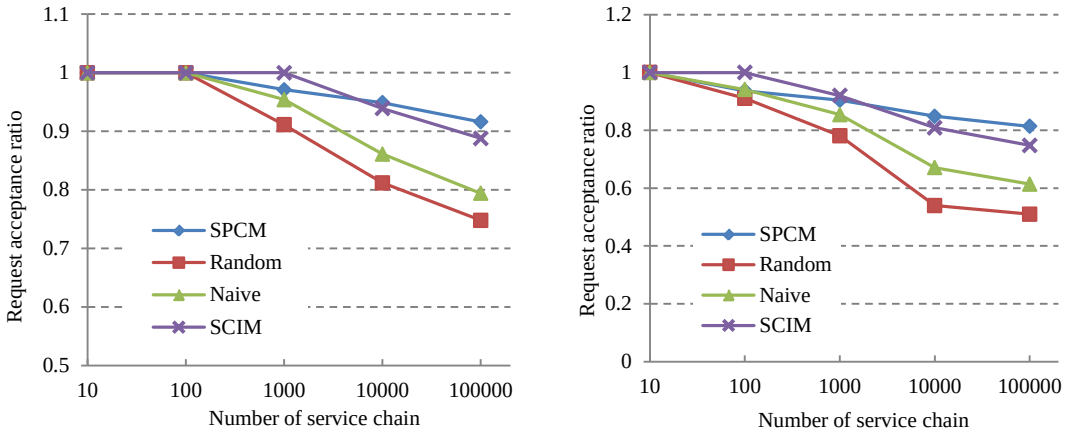
(a) RUR results on the real topology (b) RUR results on the synthetic topology

Fig. 7. The results of resources utilization ratio on two topologies

C. Comparison of different methods

In this section, we compare our SPCM algorithm with other three algorithms, denoted as “Random”, “Naive” and “SCIM” respectively. Here, “Random” way means randomly selecting nodes and links for service path of each service chain. “Naive” approach simulates a manual service chain placement which installs each service chain upon a node, and after the current node’s resource is exhausted, the next node is started. “SCIM” proposed by [18] uses the simulated annealing algorithm to find the service path. The evaluation results of different methods in the real topology are shown in **Fig. 8**, **Fig. 9** and **Fig. 10.** And the computing results for “Naive” approach can be chosen as the performance benchmark.

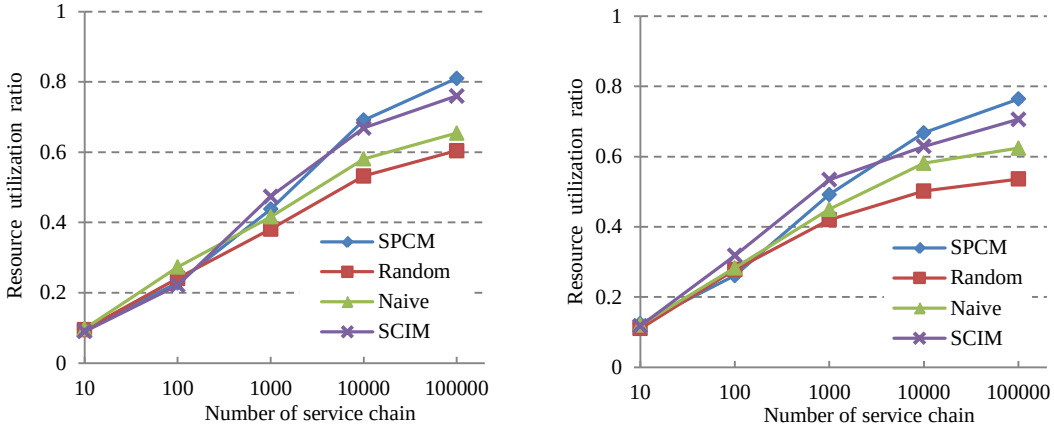
Firstly, **Fig. 8** shows the results of *RAR* with different service chain length. When the increase of number of service chains makes the network overload, *RAR* continues to decrease. SPCM and SCIM can always outperform than other two methods, and SPCM performs better than SCIM when the number of service chains is greater than 10^4. The reason is that SPCM can globally allocate network resources so that the service chain requests are able to be accepted as much as possible. With the comparison of **Fig. 8**(a) and **Fig. 8**(b), we can obtain that the longer the service chains are, the more obvious the advantage of our SPCM method is.



(a) RAR results with Length_C=2 (b) RAR results with Length_C=4

Fig. 8. The results of request acceptance ratio with different methods

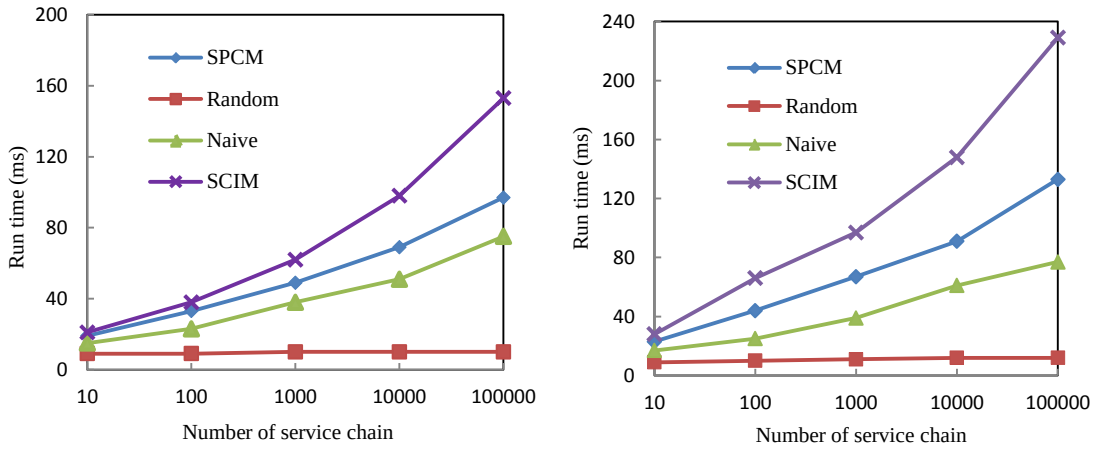
Secondly, the results of *RUR* with different service chain length are illustrated in **Fig. 9**. Since SPCM and SCIM can cooperatively manage and schedule the network resources, they can achieve approximate performance on *RUR* and lead to higher resources utilization than other two approaches. The improvement of resources utilization ratio caused by SPCM can reach more than 20%, compared to the random approach which is generally difficult to satisfy the service chain requests. This is because that with the increase of the number of service chains, SPCM algorithm can rationally deploy the service chains by overall consideration of all network node resources, while the random method is more dependent on the resources of a single node.



(a) RUR results with Length_C=2 (b) RUR results with Length_C=4

Fig. 9. The results of resources utilization ratio with different methods

Thirdly, the runtime reflects the response time from the request to the service chain construction completion and also show the computation overhead of different methods. In the graph on **Fig. 10**, we measure the required run time by each approach with respect to an increasing number of service chains. In **Fig. 10**, we can see that the runtime of random method is the least and the size remains basically the same. The reason is that the random strategy completes the construction request based on the random number, but does not need to solve the optimization problem. The running time of the other three methods grows with the number of service chains, and SCIM method presents the most running time. Our SPCM method saves average 30% running time while achieving approximate service efficiency compared to SCIM algorithm. Although the runtime of SPCM is longer than the Naive algorithm, RAR and RUR of the Naive algorithm are worse.



(a) Runtime results with Length_C=2 (b) Runtime results with Length_C=4

Fig. 10. The results of runtime with different methods

5.3 Prototype Implementation

Towards dependable validation of CNSC mechanism, we design a proof-of-concept implementation as shown in **Fig. 11**. In the prototype, CNSC module and the SR module are operated on the SDN controller which deploys service chaining decisions through south interface. We use ONOS as the SDN controller and Openflow1.3 protocol as the south interface between the controller and switches. ClickOS platforms are used to simulate virtual atomic service instances in the network.

The prototype is a mesh network that consists of five OpenFlow switches implemented by NetFPGA-10G, and four servers supporting three ClickOS platforms and ONOS controller are connected with switches. The testing scenario reflects a typical broadband network deployment case which describes the communication between users and the video server.

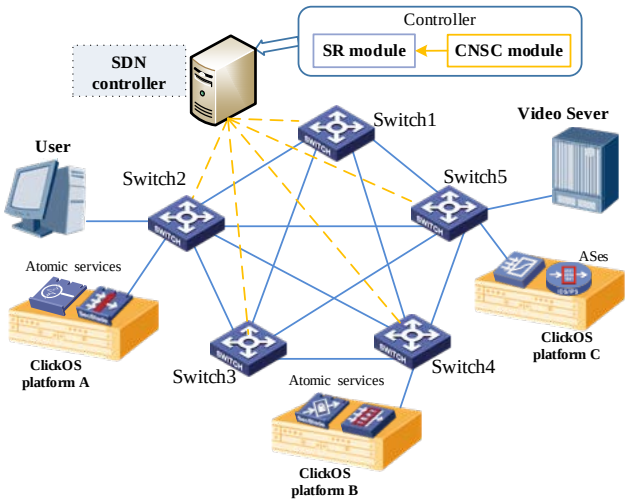
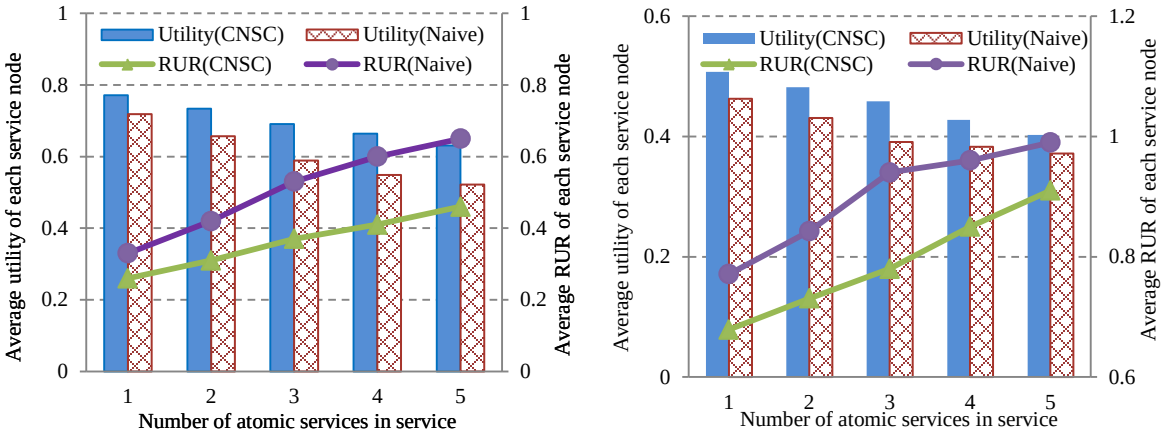


Fig. 11. The illustration of CNSC prototype.

In order to show the usability of our prototype in the real network, we setup the service scene where the user sends a message to get the files stored in the server and requires a set of services. We simulate the network traffics based on datacenter network traffic traces. Then a robot program is developed to generate a number of service chains policies for traffics, and each service chain has different the number of atomic services which increase from 1 to 5. We calculate average RUR and average utility values of each node through Eq.(11) and compare our method with the Naive approach, as shown in **Fig. 12**.

We get that the average utility values of network node from CNSC are about 0.71 and 0.43 which are respectively calculated by meaning the values of blue bars in **Fig. 12(a)** and **Fig. 12 (b)**, while the average utility values of network node from Naive approach only are 0.58 and 0.37, which are respectively calculated by meaning the values of red bars in **Fig. 12(a)** and **Fig. 12 (b)**. Therefore, our CNSC can improve the node utility 15% compared to

Naive approach. This is because our SPCM orchestrates service chains cooperatively in the network and could fully utilize the resource on different service nodes.



(a) Total service chain number on the fly 10^2 (b) Total service chain number on the fly 10^4

Fig. 12. The average utility value of network node under different service chain number

6.Conclusions

This paper has presented a configurable network service chain (CNSC) mechanism to provide network services in a flexible way and implement a scheme prototype by combining SDN controller with NFV platforms. We first abstract the network service functions as atomic services and formulate the problem of network service chaining. Then a network service chaining framework is proposed to satisfy the service policy requests by cooperatively combine atomic service instances in the optimal way. In this framework, we design service path computation algorithm (called SPCM) based on a service capacity matrix of node and a link mapping with segment routing. Finally, we validate the performance of our SPCM algorithm in an experimental environment. And with SPCM solver as the core, we design a prototype system to demonstrate the functionality and advantages of CNSC architecture. In the future work, we will increase the service prediction mechanism and the network state sensing mechanism, so that our model can be extended to adapt to dynamical construction of service chains.

References

[1] S. W. Ahn, S. H. Lee, S. H. Yoo, D. Y. Park, D. Kim, C. Yoo, “Isolation schemes of virtual network platform for cloud computing,” *KSII Transactions on Internet and Information Systems*, vol. 6, no. 11, pp. 2764- 2783, 2012. Article (CrossRef Link)

[2] M. Chiosi, D. Clarke, P. Willis, et al., “Network functions virtualisation –introductory white paper,” *SDN and OpenFlow world congress, Darmstadt*, Germany, 2012. Article (CrossRef Link)

[3] Open Networking Foundation, "Software-defined networking: the new norm for networks," white paper, April, 2012. Article (CrossRef Link)

[4] P. Quinn, T. Nadeau, "Problem statement for service function chaining," RFC 7498, 2015. Article (CrossRef Link)

[5] W. Lee, Y. H. Choi, N. Kim, "Study on virtual service chain for secure software defined networking," *Advanced Science and Technology Letters*, vol. 29, pp.177-180, 2013. Article (CrossRef Link)

[6] P. Quinn, J. Guichard, R. Fernando, et. al, "Network service header," Internet-Draft, draft-ietf-sfc-nsh-01.txt, IETF, 2014. Article (CrossRef Link)

[7] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. of the ACM SIGCOMM'13*, Hong Kong, China, pp. 27-38, August 12-16, 2013. Article (CrossRef Link)

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no.2, 2008. Article (CrossRef Link)

[9] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvret, R. Manghirmalani, R. Mishra, "StEERING: a software-defined networking for inline service chaining," in *Proc. of the 21st IEEE International Conference on Network Protocols (ICNP)*, pp.1-10, 7-10 Oct., 2013. Article (CrossRef Link)

[10] S. K. Fayazbakhsh, V. Sekar, M. Yu and J. C. Mogul, "FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proc. of the HotSDN*, pp. 19-24, 2013. Article (CrossRef Link)

[11] A. Gember-Jacobson, R. Viswanathan, C. Prakash, et al., "OpenNF: enabling innovation in network function control," in *Proc. of the ACM SIGCOMM'14*, Chicago, 2014. Article (CrossRef Link)

[12] M Xia, M Shirazipour, Y Zhang, et al., "Optical service chaining for network function virtualization," *IEEE Communications Magazine*, vol. 53, no.4, pp. 152-158, 2015. Article (CrossRef Link)

[13] L. Sun, H. Dong, J. Ashraf, "Survey of service description languages and their issues in cloud computing," *Eighth International Conference on Semantics, Knowledge and Grids (SKG)*, pp.128-135, October 2012. Article (CrossRef Link)

[14] C. Monsanto, J. Reich, N. Foster, J. Rexford and D. Walker, "Composing software-defined networks," in *Proc. of USENIX NSDI*, 2013. Article (CrossRef Link)

[15] S. Shin, P. Porras, V. Yegneswaran, et al., "FRESCO: modular composable security services for software-defined networks," in *Proc. of NDSS*, 2013. Article (CrossRef Link)

[16] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco and F. Huici, "ClickOS and the art of network function virtualization," NSDI, 2014. Article (CrossRef Link)

[17] A. Baumgartner, V. S. Reddy, T. Bauschert, "Mobile core network virtualization: a model for combined virtual core network function placement and topology optimization," in *Proc. of 1st IEEE Conference on Network Softwarization (NetSoft)*, London, pp. 1-9, 2015. Article (CrossRef Link)

[18] G Z Cheng, H C Chen, H C Hu, et al., "Enabling network function combination via service chain instantiation," *Computer Netwoks*, pp.396-407, 2015. Article (CrossRef Link)

[19] P. Wang, J. Lan, X. Zhang, Y. Hu, S. Chen, "Dynamic function composition for network service chain: model and optimization," *Computer Networks*, vol.92, pp.408-418, 2015. Article (CrossRef Link)

[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 Communications*, vol. 22, no. 6, pp. 15-23, 2015. Article (CrossRef Link)

[21] T. X. Li, H. C. Zhou, H. B. Luo, "A new method for providing network services: Service function chain," *Optical Switching and Networking, available online*, 30 September 2015. Article (CrossRef Link)

[22] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proc. of SIGCOMM'15*, August 2015. Article (CrossRef Link)

[23] S. Liu, W. Jia, "An adaptive virtual machine location selection mechanism in distributed cloud," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 12, pp. 4776-4798, Dec. 2015. Article (CrossRef Link)

[24] R. Hartert, P. Schaus, S. Vissicchio and O. Bonaventure, "Solving segment routing problems with hybrid constraint programming techniques," *CP 2015*, Cork, 2015. Article (CrossRef Link)

[25] The Open Network Operating System (ONOS). Article (CrossRef Link)

[26] C. Fils_ls et al., "Segment routing architecture," *Internet draft*, IETF, 2014. Article (CrossRef Link)

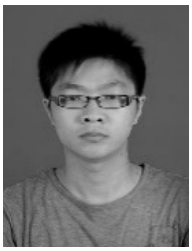
[27] Internet2 open science, scholarship and services exchange." Article (CrossRef Link)

[28] E. W. Zegura, K. L. Calvert, S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM*, vol. 2, pp. 594-602, 1996. Article (CrossRef Link)

Gang Xiong received his B.E. and M.E. degrees in 2009 and 2012, respectively. He is currently a Ph.D. candidate in National Digital Switching System Engineering and Technological R&D Center (NDSC). His research interests are future network and network security. Email: xg1226@126.com

Yuxiang Hu is an assistant professor in NDSC, China. His research interests mainly include network security, routing protocols and future network.

Julong Lan is the full Professor in NDSC, China. His research interests mainly include routing and switching design, routing protocols, resource scheduling, network security, and future network.



Guozhen Cheng received his Ph.D. degree in 2015 and currently is a lecturer at the NDSC. His research interests include Internet architecture, network security, software-defined networking and network function virtualization.