Central Control over Distributed Service Function Path

Dan Li[1*], Julong Lan[1] and Yuxiang Hu[1]
[1] Institute of Information Technology, PLA Strategic Support Force Information Engineering University
Zhengzhou 450000 - China
[e-mail: pkulidan@foxmail.com]
*Corresponding author: Dan Li

Received January 2, 2019; revised July 10, 2019; revised August 1, 2019; accepted September 15, 2019; published February 29, 2020

Abstract

Service Function Chaining (SFC) supports services through linking an ordered list of functions. There may be multiple instances of the same function, which provides a challenge to select available instances for all the functions in an SFC and generate a specific Service Function Path (SFP). Aiming to solve the problem of SFP selection, we propose an architecture consisting of distributed SFP algorithm and central control mechanism. Nodes generate distributed routings based on the first function and destination node in each service request. Controller supervises all of the distributed routing tables and modifies paths as required. The architecture is scalable, robust and quickly reacts to failures because of distributed routings. Besides, it enables centralized and direct control of the forwarding behavior with the help of central control mechanism. Simulation results show that distributed routing tables can generate efficient SFP and the average cost is acceptable. Compared with other algorithms, our design has a good performance on average cost of paths and load balancing, and the response delay to service requests is much lower.

Keywords: Service Function Chaining, Network Function Virtualization, function instantiation, distributed routing, central control

This version propose an architecture consisting of distributed routing algorithm and central control mechanism to solve the problem of service function path selection. This research was supported by The National Nature Science Foundation of China (Grant No. 61521003), The National Natural Science Foundation of China (61702547) and The National Key Research and Development Program of China (2017YFB0803204).

1. Introduction

With the rapid development and wide use of Internet, the store-and-forward function is too simple to satisfy the requirement of emerging applications. Network Function Virtualization (NFV) [1], proposed in 2012, separates virtual network functions from hardware or proprietary devices and further initiatively adapts functions to the specific application environment. Under the paradigm of NFV, Service Function Chaining (SFC) [2] supports services through linking an ordered list of functions (such as firewalls, proxies, Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs), which are used to be realized as middleboxes [3] on virtualization platforms.

There may be multiple instances of the same function, which provides a challenge to select available instances for all the functions in an SFC and generate a specific Service Function Path (SFP) [4]. Most of existing works mainly target on generating SFP in a centralized control architecture because the calculation of best SFP is too heavy for distributed nodes. What is more, it is much easier to balance load or bandwidth in the whole network with the help of central control. However, as the length of the service function chain increases, the number of paths that meet the service request increases exponentially. It is also a great burden on controller to calculate all of the best SFPs in a large network. In addition, the network environment and physical topology have the probability of change in the actual network. If network parameters change or nodes and links are added or deleted in the physical topology, controller needs to consume a lot of resources and time to recalculate SFPs and forward control messages, which restricts the scalability and robustness of network. Therefore, both centralized and distributed algorithms have their own shortcomings, so we introduce an architecture combining a distributed routing table generation algorithm for SFP and a central control mechanism to combine the advantages of distributed and centralized algorithms.

In this paper, we study the problem of SFP selection, and introduce an architecture combining a distributed routing table generation algorithm for SFP and a central control mechanism, called as Central-Distributed Service Function Path (CDSFP). There are two routing tables in each node, distributed and centralized routing tables. Distributed SFP algorithm is just like traditional link-state protocols such as OSPF [5] and IS-IS [6], where nodes compute shortest paths over a synchronized view of network topology. However, to support SFP selection in the proposed algorithm, not only destination nodes but also the first functions in service requests are in distributed routing tables. Controller supervises all of the distributed routing tables. If there is a path needing modification, controller calculates best SFP and sends centralized routing messages to nodes on the path as required. Centralized routing tables have a higher priority than distributed routing tables, so that the distributed path can be modified. CDSFP is scalable, robust and quickly reacts to failures because of distributed routings, and enables centralized and direct control of the forwarding behavior with the help of central control mechanism.

In summary, we make the major contributions in this paper as follows:

1) A distributed path can connect function instances and destination node in sequence, so it must be an effective path corresponding to the service request. To the best of our knowledge, this work is the first effort that uses distributed routing tables to solve the problem of SFP selection.
2) In most networks, best SFP is not necessary if the cost of path is acceptable and other service requirements can be satisfied. By setting reasonable parameters in the proposed

distributed algorithm, the average cost of paths is just a little higher than best SFP, which can satisfy the requirements of most services.

3) Central control mechanism can reduce the average cost of SFPs, and consider many restrictive conditions synthetically, such as bandwidth limitations of links and load balancing of the whole network.

The rest of this paper is organized as follows. Related works are given in Section 2. Section 3 formulates SFP selection problem and introduces our proposed architecture. In Section 4 we describe distributed SFP algorithm and analyze its performance. Section 5 gives a central control mechanism to modify distributed paths. In Section 6 simulation results of the proposed central-distributed approach are presented and we compare its performance with other existing solutions. Finally, this paper is summarized in Section 7.

2. Related Work

There is already some works on SFC so far and most researches aim to design algorithms in a centralized control architecture. Authors in [7] consider link delays and computational delays and set up the problem of deploying service function chains over network function virtualized architecture as an integer linear programming optimization problem, along with important constraints such as total deployment costs and service level agreements. Ref [8] proposes a unified control and optimization framework that combines SFC and SDN. By modeling the network and services together and developing optimization techniques, this framework benefits the overall system performance in various scenarios. In [9], authors develop an approach that uses three algorithms in VNF placement, SFC routing, and VNF migration in response to changing workload. The objective is to minimize the rejection of SFC bandwidth and consolidate VNFs in as few servers as possible so as to reduce the energy consumed. Ref [10] studies how to optimize SFC deployment and readjustment in the dynamic situation. It tries to jointly optimize the deployment of new users' SFCs and the readjustment of in-service users' SFCs while considering the trade-off between resource consumption and operational overhead. Segment routing is used for optimal link mapping in [11] and [12] to promote request acceptance ratio and resources utilization ratio. In [13], authors study the joint problem of service function chain deploying and path selection for bandwidth saving and VNF reuse. They describe the problem as a multi-objective and multi-restriction problem, and propose a heuristic service function chain deployment algorithm. The placement aspect of SFC is addressed both in [14] and [15] by finding the best locations and hosts for the functions while respecting user requirements and maximizing provider revenue. Ref [16] illustrates a dynamic service function composition model which optimizes cost, load and other QoS metrics simultaneously and proposes a polymorphic derivation algorithm based on Markov approximation to obtain an optimal solution. Authors in [17] propose a novel resource allocation architecture which enables energy-aware SFC for SDN-based networks, considering also constraints on delay, link utilization, server utilization. They formulate the problems as integer linear programming optimization problems and design a set of heuristic to find near-optimal solutions in timescales suitable for practical applications.

There are some distributed algorithms that can optimize delay, expense or some other QoS metrics. An adaptive triggering policy based on link-usage statistics, is proposed in [18] in order to reduce the volume of link state update traffic without deterioration of QoS. The algorithms in [19] and [20] define their cost function of links which is a weighted sum of response time, cost, reliability and availability and generate best paths based on the shortest path algorithm. Central control is introduced in [21] and [22] to modify distributed paths generated according to OSPF to regulate the bandwidth, load or other QoS metrics. However, distributed paths generated in the above algorithms cannot correspond to SFCs because data packets are forwarded only according to destination nodes in routing tables. SpiderNet in [23] is a P2P service composition framework that can simultaneously guarantee QoS and load balancing in distributed environment. Authors in [24] propose an algorithm for QoS assurance and load balancing (QALB) on the basis of SpiderNet by changing the weights of cost function dynamically. In [23] and [24], the service requests and responses are generated in distributed nodes, but the selection of path is made centrally by sink node.

3. Problem Description

The symbols with their definitions used in this paper are shown in **Table 1**. We model a network as an edge-weighted graph $G=(V,E)$ in which V and E means the set of nodes and links, and c_e is the cost of each link $e \in E$. In this paper, the cost of link can be defined as delay, energy, expense or some other parameters needing to be reduced. There are some kinds of functions. Each function F_i may have a set of instances $\{F_i^1, F_i^2, \dots, F_i^{M_i}\}$ deployed on different nodes, where M_i is the number of instances.

Table 1. List of symbols used in this paper

Notation	**Description**
V	The set of nodes
E	The set of links
c_e	The cost of link e
L_v	The maximum available load of node v
B_e	The maximum available bandwidth of link e
F	The set of functions
n	The number of types of functions
$F_i^{j_i}$	The $j_i th$ instance of function F_i
k	The length of SFC
$C(P)$	The cost of path P
P_i	The SFP selected for the ith service request
l_i	The required load for the ith service request
l_v	The present load of node v
b_i	The required bandwidth for the ith service request
AD	Paths from A to D
$AF_1 \dots F_k D$	Paths connecting the instances of k functions sequentially from A to D
d	The upper bound of the cost of the shortest paths between any nodes
h	The number of nodes
p	The number of instances
h_1	The average number of hops of the shortest paths between any nodes
$f(k)$	The expectation of $\min C(F_1F_2)+\dots+\min C(F_{k-1}F_k)+\min C(F_kD)$
α	The weight in the cost function with service

β_1	The parameter to set a threshold about cost of paths
β_2	The parameter to set a threshold about load balancing

Let R be a service request, $R=\{A,F_1,F_2,...,F_k,D\}$, in which k is the number of required functions, A is source node, D is destination node, and $\{F_1,F_2,...,F_k\}$ is SFC meaning the logical ordered list of functions in a service request. We assume that the functions in each SFC are different from each other, and different service requests can share the same instance. The SFP selection problem is to select one instance for each function in R, composing a service response $\{F_1^{j_1},F_2^{j_2},...,F_k^{j_k}\}$ to satisfy R, where $F_i^{j_i}$ means the $j_i th$ instance of F_i.

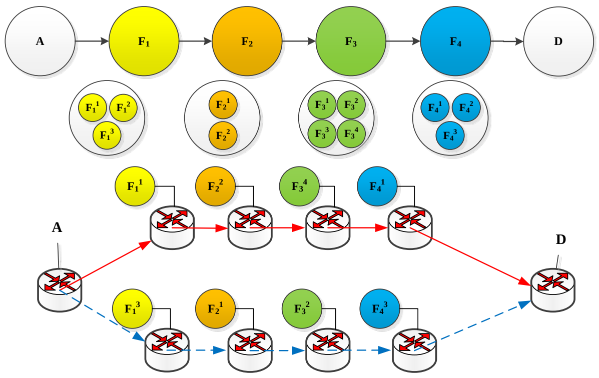

Fig. 1. An example of SFP selection problem

Fig. 1 is an example of SFP selection problem. When source node receives a service request $R=\{A,F_1,F_2,F_3,F_4,D\}$, the red solid path $\{F_1^1,F_2^2,F_3^4,F_4^1\}$ and the blue dashed path $\{F_1^3,F_2^1,F_3^2,F_4^3\}$ both can satisfy the service request. There may be any service request at any time with no prior information. The fundamental objective of SFP selection is to find a path which has low cost and satisfies the constraint condition simultaneously whenever a service request appears. Given a set of service requests, this objective can be mathematically described as:

$$\text{Minimize} \sum_i C(P_i) \tag{1}$$

Subject to:

$$\sum_{v_p \in P_i} l_i \le L_v, \forall v_p \in V \tag{2}$$

$$\sum_{(v_p,v_q)\in p_i} b_i + \sum_{(v_q,v_p)\in p_i} b_i \le B_e, \forall (v_p,v_q) \in E \tag{3}$$

Eq. (1) means to minimize the cost of paths. Eqs. (2) and (3) are load and bandwidth constraint conditions. To find a feasible solution which can approach the objective, we propose an architecture combining a distributed routing table generation algorithm for SFP selection and a central control mechanism.

4. Distributed SFP Selection Algorithm

4.1 Structure of Routing Tables

In traditional link-state protocols, there are only destination nodes in routing tables. However, in the problem of SFP selection, both functions and destination nodes in service requests should be considered. The problem is that if all of the possible service requests are put into routing tables, the number of routings will increases dramatically with the number of functions in network. As a result, the complexity to generate and maintain routing tables is extremely high, and the routing convergence time is too long to make sure there is no routing loop or data loss. Based on the above analysis, we just put destination node and the first function in a service request into a distributed routing. Once some node provides the first function, it is removed from the service request and replaced by the next function. Each node only needs to decide whether to provide the first function in the immediate request without considering previous functions. For example, if node B receives $R=\{A,F_1,F_2,F_3,F_4,D\}$ and provides the first function F_1, the request becomes $R=\{A,F_2,F_3,F_4,D\}$ and the first function becomes F_2. In this way, a distributed path can connect function instances and destination node in sequence, so it must be an effective path corresponding to the service request.

Consider a network with seven nodes shown in **Fig. 2**, where solid lines connect neighbors. B and E can provide function F_1 while C and D can provide function F_2. Part of the routing table of A expressed beside A in **Fig. 2** is shown in **Table 2**, where the first function is shown as 0 when there is no functions in service requests. The routing tables of B and C are also shown in the figure. No matter how many functions there are in a service quest, the nodes just make decisions according to the first function and destination node.

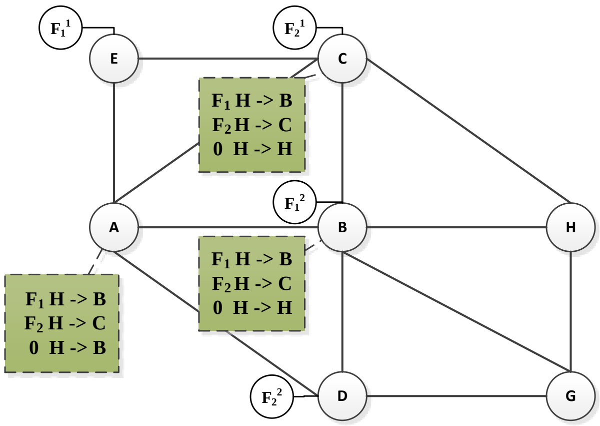

Fig. 2. Example of distributed routing tables

Table 2. Part of distributed routing table of A

First function	Destination node	Next hop
F_1	H	B
F_2	H	C
0 (without service)	H	B

4.2 Cost Function

The core of distributed SFP algorithm is cost function, based on which the cost of paths is calculated and compared through different neighbor nodes. Because both functions and destination nodes are considered in the proposed algorithm, we divide paths into two categories, (1) paths with service and (2) paths without service. The cost of paths without service is as same as the shortest path algorithm. The cost of the shortest path from A to D without service is defined as $\min C(AD)$.

Considering a service request $\{A, F_1, F_2, ..., F_k, D\}$, the cost of the shortest SFP can be accurately expressed as

$$C(AF_1...F_kD) = \min C(AF_1) + \min C(F_1F_2) + ... + \min C(F_{k-1}F_k) + \min C(F_kD) \tag{4}$$

In the proposed distributed SFP algorithm, there is only the first function in a routing, so it is usually not possible to calculate and compare the cost by (4). Therefore, we have to compare the cost of paths with service by $\min C(AF_1)$ and $\min C(F_1D)$. Because $\min C(AF_1)$ is an addend in (4), we need to analyse the relationship between $\min C(F_1D)$ and $\min C(F_1F_2) + ... + \min C(F_{k-1}F_k) + \min C(F_kD)$, which is related to the length of SFC.

Firstly, we consider the situation that there are just two functions, F_1 and F_2. Without loss of generality, assuming that the cost of the shortest path between any two nodes is not more than d, we can get,

$$0 \le \min C(F_1F_2), \min C(F_2D) \le d \tag{5}$$

According to the properties of inequality and triangle, the relationship can be obtained as follows based on (5)

$$\min C(F_1D) \le \min C(F_1F_2) + \min C(F_2D) \le 2d \tag{6}$$

The relationship between $\min C(F_1F_2) + \min C(F_2D)$ and $\min C(F_1D)$ is according to the position of instances of F_2. Obviously, if at least one node on the path of $\min C(F_1D)$ can provide F_2, the minimum of $\min C(F_1F_2) + \min C(F_2D)$ is equal to $\min C(F_1D)$. Let p be the number of instances of function F_2, h be the number of all the nodes, and h_1 be the number of hops of $\min C(F_1D)$. The probability that there is at least one instance of F_2 on the path of $\min C(F_1D)$ is $1-(h-h_1/h)^p$, which is shown in (7).

$$P\left(\min\left(\min C(F_1F_2) + \min C(F_2D)\right) = \min C(F_1D)\right) = 1 - \left(\frac{h-h_1}{h}\right)^p \tag{7}$$

On the contrary, if none of the instances are on the path of $\min C(F_1D)$, $\min C(F_1F_2) + \min C(F_2D)$ is scattered randomly between $\min C(F_1D)$ and $2d$. Assuming that the instances are evenly distributed in a two-dimensional plane, we can estimate the minimum of $\min C(F_1F_2) + \min C(F_2D)$ in (8), which happens with probability $(h-h_1/h)^p$

$$P\left(\min\left(\min C(F_1F_2) + \min C(F_2D)\right) = \left(\frac{2d - \min C(F_1D)}{\sqrt{p+1}} + \min C(F_1D)\right)\right) = \left(\frac{h-h_1}{h}\right)^p \tag{8}$$

Combining (7) and (8), $\min\left(\min C\left(F_1F_2\right)+\min C\left(F_2D\right)\right)$ is in accord with the Bernoulli distribution. Therefore, the expectation of $\min\left(\min C\left(F_1F_2\right)+\min C\left(F_2D\right)\right)$, which is represented as $f\left(2\right)$, can be calculated as

$$\begin{aligned} f\left(2\right) &= \left(1-\left(\frac{h-h_1}{h}\right)^p\right)\min C\left(F_1D\right)+\left(\frac{h-h_1}{h}\right)^p\left(\frac{2d-\min C\left(F_1D\right)}{\sqrt{p+1}}+\min C\left(F_1D\right)\right) \\ &= \left(\frac{h-h_1}{h}\right)^p\left(\frac{2d}{\sqrt{p+1}}\right)+\left(1-\left(\frac{h-h_1}{h}\right)^p\frac{1}{\sqrt{p+1}}\right)\min C\left(F_1D\right) \end{aligned} \tag{9}$$

To make following analysis easier, we rewrite $f\left(2\right)$ as (10). Which means inserting F_2 into $f\left(1\right)$

$$\begin{aligned} &f\left(2\right)=a+b\times f\left(1\right) \\ &\text{where } a=\left(\frac{h-h_1}{h}\right)^p\left(\frac{2d}{\sqrt{p+1}}\right), b=\left(1-\left(\frac{h-h_1}{h}\right)^p\frac{1}{\sqrt{p+1}}\right), f\left(1\right)=\min C\left(F_1D\right) \end{aligned} \tag{10}$$

Secondly, we incrementally increase the length of SFC. We use $f\left(3\right)$ to represent $\min\left(\min C\left(F_1F_2\right)+\min C\left(F_2F_3\right)+\min C\left(F_3D\right)\right)$ which means inserting F_3 into $f\left(2\right)$, so that $f\left(3\right)$ can be deduced as

$$f(3)=a+bf\left(2\right) \tag{11}$$

By analogy, we can obtain the recursion formula of $f\left(k\right)$, which means the expectation of $\min C\left(F_1F_2\right)+...+\min C\left(F_{k-1}F_k\right)+\min C\left(F_kD\right)$.

$$f\left(k\right)=a+bf\left(k-1\right) \tag{12}$$

Substituting $f\left(1\right)=\min C\left(F_1D\right)$ into (12), we can express $f\left(k\right)$ as

$$f\left(k\right)=a+ab+...+ab^{k-2}+b^{k-1}\min C\left(F_1D\right)=\sum_{i=0}^{k-2}ab^i+b^{k-1}\min C\left(F_1D\right) \tag{13}$$

Finally, substituting (13) into (4), we have

$$C(AF_1...F_kD)=\min C\left(AF_1\right)+f\left(n\right)=\min C\left(AF_1\right)+b^{k-1}\min C\left(F_1D\right)+\sum_{i=0}^{k-2}ab^i \tag{14}$$

The third addend is a constant which has no effect on the comparison of cost, so we compare the cost of paths with service by $\min C\left(AF_1\right)+\alpha\min C\left(F_1D\right)$, where

$$\alpha=b^{k-1}=\left(1-\left(\frac{h-h_1}{h}\right)^p\frac{1}{\sqrt{p+1}}\right)^{n-1} \tag{15}$$

In a stabilizing network, parameters such as h, h_1 and p are almost invariable. The parameter k should be the average length of SFCs in the network, by which the weight α can be calculated exactly to reduce the average cost in the network.

4.3 Routing Update

A node cannot generate its routing table by itself, so we should make the strategy to share and update routings between neighbor nodes. Each node should share its routings to neighbor nodes periodically and update its routing table as soon as receiving routing messages. Like the cost of paths, routing messages also can be divided into two categories, (1) routings without service and (2) routings with service. Some routing messages shared by *A* are shown in **Table 3**, in which $C(A0H)$ has no function, and the others have F_1 or F_2 as the first function.

Table 3. Some routing messages *A* shares

First function	Destination node	Routing message
F_1	H	$C(AF_1H)$
F_2	H	$C(AF_2H)$
0 (without service)	H	$C(A0H)$

Once a node receives a routing message with service, it adds the cost between it and the neighbor node to the cost in routing message, and compares the result with other paths containing the same function and destination node. The first two lines of **Table 4** illustrate the update when *E* receives routing messages with service from *A*.

Once a node receives a routing message without service, it adds the cost between it and the neighbor node to the cost in routing message, and compares the result with other paths without service, as shown in line 3 of **Table 4**. What is more, if the node can provide some function F_i, it adds the cost between it and the neighbor node to the cost in routing message, multiplies the sum by α , and compares the result with other paths containing F_i. In the last line of **Table 3** we can see that, because *E* can provide F_1, a new routing with F_1 needs to be added when a routing message without service $C(A0H)$ is received.

Table 4. Routing update when *E* receives routing messages.

Routing message	First service	Destination node	Routing cost
$C(AF_1H)$	F_1	H	$C(AE)+C(AF_1H)$
$C(AF_2H)$	F_2	H	$C(AE)+C(AF_2H)$
$C(A0H)$	0 (without service)	H	$C(AE)+C(A0H)$
	F_1	H	$\alpha(C(AE)+C(A0H))$

Algorithm 1 is designed as the pseudo-code of routing update. After routing update and comparison of costs, nodes should put routings with the lowest cost into their routing tables.

Algorithm 1: Routing update (for example, *E* receives routing messages from *A*)

```
1.  for each routing message from $A$ arrived
2.      set $c_A$ = cost in routing message
3.      set $c_{AE}$ = cost between $A$ and $E$
4.      if there is no service in the message
5.          $c_E$ without service = min( $c_A + c_{AE}$ , $c_E$ without service)
6.          if this node can offer function $F_i$
7.              $c_E$ with $F_i$ = min( $\alpha(c_A + c_{AE})$ , $c_E$ with $F_i$ )
8.      else  // there is function $F_i$ in the message
9.          $c_E$ with $F_i$ = min( $c_A + c_{AE}$ , $c_E$ with $F_i$ )
10.     end if
```

```
11.   if $c_E$ has been changed
12.     send $c_E$ to other neighbor nodes and controller
13.   end if
14. end for
```

Convergence time is an important indicator of distributed routing update, which means the time from detection of changes in network to the end of routing update. To analyze convergence time, we refer to the shortest path algorithm of OSPF. Considering a network with h nodes and n functions, the convergence time of OSPF is proportional to the size of routing table ($T_{OSPF} = O(h)$) [25]. In our proposed algorithm, according to **Table 1,** the distributed routing table consists of a OSPF routing table without service and n OSPF routing tables with service, which means $(n+1)h$ routings in each table. so the convergence time is only $T_{DSFP} = O((n+1)h)$. Because n is usually a single digit, T_{DSFP} can satisfy the communication requirement in many networks.

4.4 Routing Lookup

After the generation and update of routing tables, nodes can forward or process data according to service requests, as shown in Algorithm 2. When a data packet arrives, the node gets the first function and destination node (line 2-3), and finds the next hop in its routing table (line 7). If the next hop is still this node, it provides the first function, deletes the first function from service request, and restarts routing lookup (line 8-12). If the next hop is another node, it sends the data packet to the next hop (line 13). The data packet is processed and forwarded again and again until it is received by destination node and SFC is empty, which means SFP ends (line 4-6).

Algorithm 2: Routing lookup

```
1.  for each data packet arrived
2.    set $F$ = the first function in service request
3.    set $D$ = destination node
4.    if $F$ is null and $D$ is this node // termination conditions of SFP
5.      keep the data
6.    else
7.      $N$ = lookup($F$,$D$) in routing table
8.      if $N$ is this node
9.        this node provide function $F$
10.       delete $F$ from service request
11.       goto (2)
12.     else // the next hop is another node
13.       send data packet to $N$
14.     end if
15.   end if
16. end for
```

Fig. 3 illustrates the forwarding process after A receives a service request $\{A, F_1, F_2, H\}$ in the network of **Fig. 2**. A looks up the next hop in its routing table according to (F_1, H) and forwards data packet to B. B looks up the next hop in its routing table according to (F_1, H), provides function F_1, and delete F_1 in the service request so that the service request becomes $\{A, F_2, H\}$. Then *B* looks up the next hop in its routing table again according to (F_2, H) and forwards data packet to *C*. *C* looks up the next hop in its routing table according to (F_2, H), provides function F_2, and delete F_2 in the service request so that the service request becomes $\{A, H\}$. Then *C* looks up the next hop in its routing table again according to $(0, H)$ and forwards data packet to *H*. Finally, the data packet is forwarded to *H* with no function needed, putting an end to this forwarding process. What need to be stressed is that *B* and *C* have to lookup routing tables twice because they provide function F_1 and F_2 respectively.

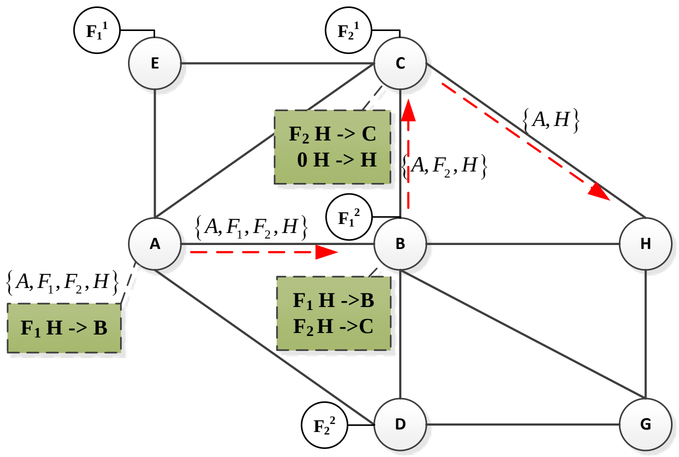

Fig. 3. Generation of distributed SFP when *A* receives a service request $\{F_1, F_2, H\}$

5. Central Control Mechanism

5.1 Strategy of Central Control

Nodes generate and update distributed routing tables, by which data can be processed and forwarded independently. However, the algorithm will not give a global optimal solution, so the cost of some routings may be very high. In addition, some routings need modification because of some restrictive condition. Therefore, we design a central control mechanism to solve the problem. To explain our algorithm clearly, in the remainder of this paper, the balance of service load, which means the number of functions provided by each node, is considered as an example of restrictive condition

Central control is realized by controller, which obtains network topology information and service load of nodes in real time. Controller also monitors the distributed routing table of each node and decides whether a distributed SFP needs modification. If so, controller calculates best SFP according to network topology and service load, and sends routing messages to nodes to change their next hops. Nodes receive and put the messages into their centralized routing tables which should be checked firstly. If there is a centralized routing corresponding to the service request, the node processes data according to it, or the node will find next hop in its distributed routing table. The difference between centralized and distributed routings is that only the first function is in a distributed routing, but the whole SFC is in a centralized routing, so that a centralized routing just changes one SFP and does not influence other paths.

If centralized routing $F_1F_2H \rightarrow E$ is send to A, new forwarding processes are illustrated in **Fig. 4**. The service request $\{A, F_1, F_2, H\}$ is the same in **Fig. 3**, but the next hop of A is changed from B to E. What is more, E and C do not need central control because their distributed routings do not need modification. If their next hops generated by distributed routings are not on the new path, controller also needs to send routing messages to them.

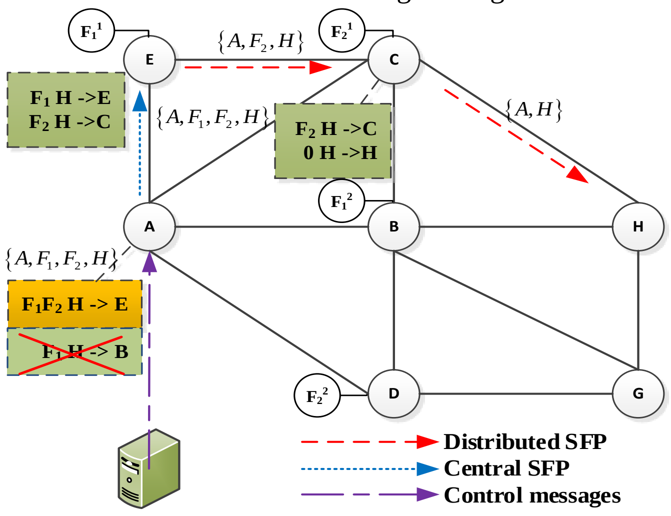

Fig. 4. Generation of SFP with central control when A receives a service request $\{F_1, F_2, H\}$

5.2 Trigger condition of Central Control

To decide whether a distributed SFP needs modification, controller needs to consider two factors. The first one is that whether the cost of that path is high, so it is necessary to design a method to obtain the cost of distributed paths and best paths. Controller monitors the distributed routing table of each node, so it can obtain the distributed SFP and calculates the cost once a service request appears. However, the calculation of best SFP for each service request will take much time. Therefore, we propose another way to estimate the cost of best SFP.

During the analysis of cost function in section 4.2, we already obtain the relationship between $\min C(F_1D)$ and $\min C(F_1F_2) + ... + \min C(F_{k-1}F_k) + \min C(F_kD)$, shown as (13). Based on that, it is easy to deduce the estimation of minimum cost from A to D with k functions, which can be expressed as (16). Distributed path needs to be changed if its cost is higher than β_1 times the cost of best SFP.

$$C_{distributed}(AF_1...F_kD) \le \beta_1 C_{best}(AF_1...F_kD)$$
$$C_{best}(AF_1...F_kD) = \sum_{i=0}^{k-1} ab^i + b^k \min C(AD) \qquad (16)$$
$$\text{where } a = \left(\frac{h-h_1}{h}\right)^p \left(\frac{4d}{\sqrt{p+1}}\right),\ b = \left(1 - \left(\frac{h-h_1}{h}\right)^p \frac{1}{\sqrt{p+1}}\right)$$

The second factor is that whether the balance of service load is satisfied. Because the objective of load balancing is to minimize the maximum of load in the network, we set a threshold about load balancing for each distributed SFP in (17), which means the maximum of load must be lower than β_2 times the average load. In addition, the value of β_1 and

β_2 influences the effect of load balancing and the proportion of central control.

$$\max(l_v) \le \beta_2 \frac{\sum_v l_v}{h} \tag{17}$$

5.3 Calculation of Best SFP

In the proposed architecture, controller does not need to work all the time, so we do not care a lot about the cost and delay of calculation. Therefore, we choose an exhaustive method with the highest calculation accuracy. Controller just traverses all the paths corresponding to the service request, sorts them by their cost, and choose a path that has the lowest cost and satisfies the balance of service load. The pseudo-code of exhaustive method is shown in Algorithm 3.

Algorithm 3: Exhaustive method

```
1.  for each service request arrived
2.    set $A$ = source node
3.    set $D$ = destination node
4.    set $S$ = SFC
5.    set $P$ = the set of paths which connect function instances of $S$ in sequence from $A$ to $D$
6.    sort $P$ by the cost of paths
7.    set $p_s$ ←the shortest path in $P$
8.    if $p_s$ does not satisfy the balance of service load
9.      delete $p_s$ from $P$
10.     go to (7)
11.   end if
12.   send routing messages to nodes along $p_s$ as required
13. end for
```

6. Experimental Classification Results and Analysis

6.1 Simulation Design

To evaluate the performance of the proposed central-distributed architecture, we design simulation scenarios in a network topology with 20 nodes that is generated randomly using Waxman-Salama model [26]. There are 5 types of functions which can be used to combine SFCs and 5 instances randomly deployed for each function in the network. Each SFC is constructed by randomly selecting functions and the length of SFC can change from 1 to 5. The source and destination of each service request are also selected randomly from all of the nodes in the network. To reduce the uncertainty of simulation, we implement 100 simulations by Monte Carlo method in each scenario and choose the average value as the simulation result.

6.2 Performance of Distributed SFP Algorithm

Firstly, we evaluate the performance of distributed SFP algorithm in the simulation scenarios above. Average cost of path is the most important performance metric of distributed algorithm. To verify whether the calculation of α is appropriate, we simulate the average cost of distributed SFP with different α in the first place. Optimal α in simulations and α calculated by (15) with different SFC length are shown in **Table 5**, in which we can see that the calculation of α agrees well with simulation results.

Table 5. Calculated weights and Optimal weights

SFC length	Calculated α	Optimal α
1	1	1
2	0.82	0.80
3	0.67	0.65
4	0.55	0.55
5	0.45	0.45

The distribution graphs of the ration of path cost to best SFP and the ration of node load to average load in 100 simulations is shown in **Fig. 5** and **Fig. 6**. We can see that the cost of about 90% distributed paths are less than 1.4 times the cost of best SFPs. Therefore, if we set $\beta_1 = 1.4$, we can make sure that the cost of all the paths can be acceptable. The distribution of load also needs to be balanced because the load of nearly 10% nodes are more than 1.5 times the average load, which should be avoided by setting $\beta_2 = 1.5$

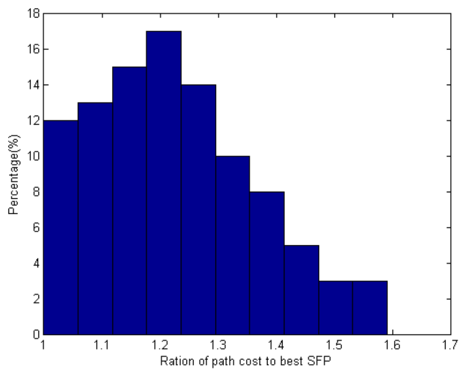

Fig. 5. Distribution graph of the ration of path cost to best SFP

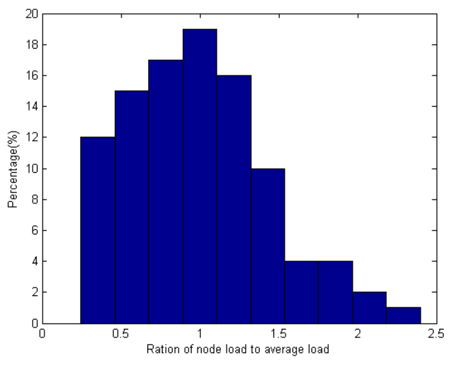

Fig. 6. Distribution graph of the ration of node load to average load

6.3 Performance of Central Control Mechanism

In this section, we compare the performance of our design (CDSFP) with other existing algorithms in distributed environment, SpiderNet in [23] and QALB in [24]. Exhaustive method is also simulated to show the performance of best paths. We set $\beta_1 = 1.4$ and $\beta_2 = 1.5$, so that the proportion of central control is about 20% (1-(1-90%)(1-90%)).

Before the comparison, we firstly validate our method to estimate the cost of best SFP in (16), because it is important to decide whether a distributed SFP needs modification in our design. The cost of best SFP and our estimation result are represented in **Fig. 7**. The estimation result is a little more or less than actual cost, which means that our estimation method is reasonable for cost judgment of distributed paths.

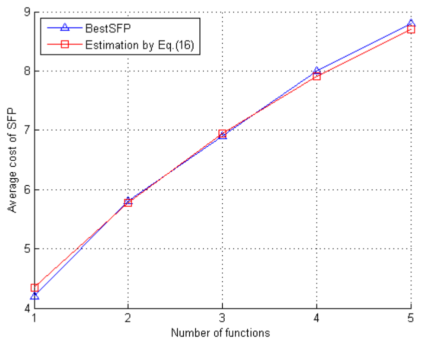

Fig. 7. Average cost of best SFP and estimation result

The maximum load in the network and the average cost of SFP are shown in **Fig. 8** and **Fig. 9** respectively. Because only 10% paths are modified by controller for load balancing, the maximum of load of CDSFP is a little higher than the other three algorithms, but the gap between CDSFP and SpiderNet is very small. However, the average cost of our proposed method is only about 10% higher than best paths given by Exhaustive method, and evidently lower than QALB and SpiderNet.

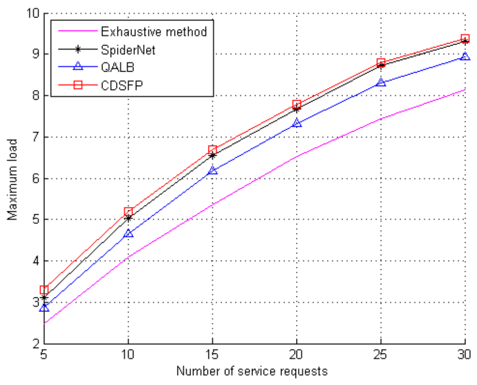

Fig. 8. The maximum load in the network using CDSFP and other three existing method

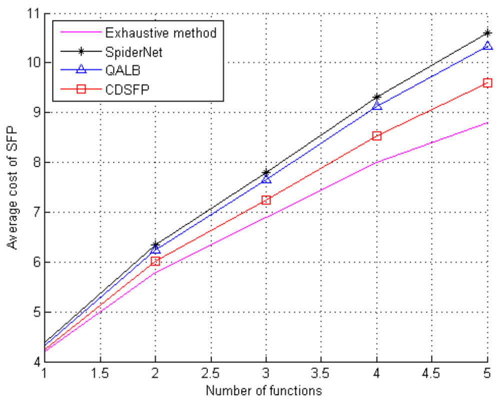

Fig. 9. Average cost of SFP using CDSFP and other three existing method

Fig. 10 shows the results on average response delay to each service request. Exhaustive method needs the longest time to calculate best paths. QALB and SpiderNet also spend much longer time than CDSFP to exchange request and response messages about each service request between nodes. In our design, distributed SFP is generated according to routing tables as soon as a service request appears, so that the response delay is mainly decided by the proportion of central control. When the proportion of central control is about 20%, the calculation time is at least 50% lower than other solutions.

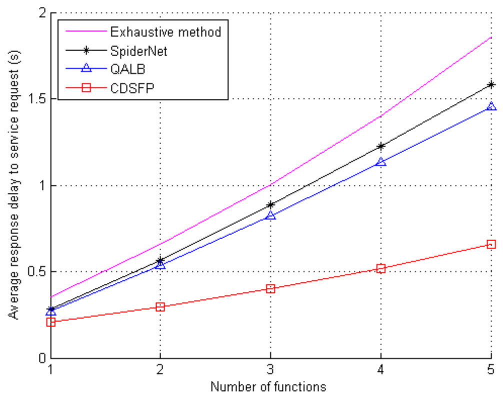

Fig. 10. Average response delay to each service request using CDSFP and other three existing method

7. Conclusion

The generation of SFP is one of the difficult problems in NFV. It is not easy to calculate the best SFP in a large network. In this paper, we propose an architecture combining distributed SFP algorithm and central control mechanism to select available instances for SFC. Nodes generate distributed routing tables and build effective paths for service requests. Controller calculates best SFP and modifies some paths by sending centralized routings, which have a higher priority. Simulation results reveal that the architecture shows a suboptimal result in the average cost .Compared with other centralized algorithms, our architecture is more scalable and robust because of distributed routings, and the average response delay to service requests is much lower. We believe that new researches can further explore this direction such as the optimization of routing tables, the method to modify paths by controller and so on.

References

[1] R Mijumbi, J Serrat, J L Gorricho, N Bouten, F De Turck and R Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236-262, Firstquarter 2016. Article (CrossRef Link)

[2] P Quinn and J Guichard, "Service function chaining: Creating a service plane via network service headers," *Computer*, vol. 47, no. 11, pp. 38-44, November, 2014. Article (CrossRef Link)

[3] X Yuan, D Huayi and W Cong, "Assuring string pattern matching in outsourced middleboxes," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no.3, pp. 1362-1375, June, 2018. Article (CrossRef Link)

[4] A M Medhat, T Taleb, A Elmangoush, G A Carella, S Covaci and T Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE*

Communications Magazine, vol. 55, no. 2, pp. 216-223, February, 2017. Article (CrossRef Link)

[5] G Apostolopoulos, R Guerin, and S Kamat, "Implementation and performance measurements of QoS routing extensions to OSPF," in *Proc. of Infocom 99 Eighteenth Joint Conference of the IEEE Computer & Communications Societies*, pp. 680-688, March21-25, 1999. Article (CrossRef Link)

[6] B Fortz and M Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756-767, May, 2002. Article (CrossRef Link)

[7] D Bhamare, M Samaka, A Erbad, R Jain, L Gupta and, H A Chan, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1-16, April, 2017. Article (CrossRef Link)

[8] 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, December, 2015. Article (CrossRef Link)

[9] V Eramo, E Miucci, M Ammar and F G Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008-2025, August, 2017. Article (CrossRef Link)

[10] J Liu, W Lu, F Zhou, P Lu and Z Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543-553, September 2017. Article (CrossRef Link)

[11] G Xiong, P Sun, Y Hu, J Lan and K Li, "An Optimized Deployment Mechanism for Virtual Middleboxes in NFV-and SDN-Enabling Network," *TIIS*, vol. 10, no. 8, pp. 3474-3497, August, 2016. Article (CrossRef Link)

[12] G Xiong, Y Hu, J Lan and G Cheng, "A Mechanism for Configurable Network Service Chaining and Its Implementation," *TIIS*, vol. 10, no. 8, pp. 3701-3727, 2016. Article (CrossRef Link)

[13] D Li, J Lan and P Wang, "Joint service function chain deploying and path selection for bandwidth saving and VNF reuse," *International Journal of Communication Systems*, vol. 31, no. 6, pp, e3523, April 2018. Article (CrossRef Link)

[14] L Wang, Z Lu, X Wen, R Knopp, R Gupta, "Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization," *IEEE Access*, vol. 4, pp. 8084-8094, November 2016. Article (CrossRef Link)

[15] M Mechtri, C Ghribi, D Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533-546, September, 2016. Article (CrossRef Link)

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

[17] M M Tajiki, S Salsano, L Chiaraviglio, M Shojafar and B Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374-388, March, 2018. Article (CrossRef Link)

[18] M Yang, T Kim, H Jung, MH Jung and SH Choi, "Performance analysis of an adaptive link status update scheme based on link-usage statistics for QoS routing," *ETRI journal*, vol. 28, no. 6, pp. 815-818, December, 2006. Article (CrossRef Link)

[19] K H Cho, "Flow Holding Time based Advanced Hybrid QoS Routing Link State Update in QoS Routing," *Journal of the Korea Society of Computer and Information*, vol. 21, no. 4, pp. 17-24, April, 2016. Article (CrossRef Link)

[20] K H Cho, " Hybrid Link State Update Algorithm in QoS Routing," *Journal of the Korea Society of Computer and Information*, vol. 19, no. 3, pp. 55-62, March, 2014. Article (CrossRef Link)

[21] S Vissicchio, O Tilmans and L Vanbever, "Central control over distributed routing," *ACM SIGCOMM Computer Communication Review*, pp.43-56, 2015. Article (CrossRef Link)

[22] M Caria, A Jukan and M Hoffmann, "SDN partitioning: A centralized control plane for distributed routing protocols," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 381-393, September, 2016. Article (CrossRef Link)

[23] X Gu, K Nahrstedt and B Yu, "SpiderNet: An integrated peer-to-peer service composition framework," in *Proc. of 13th IEEE International Symposium on High performance Distributed Computing*, pp. 110-119, June 4-6, 2004. Article (CrossRef Link)
[24] K Lee and S Park, "A service path selection and adaptation algorithm for QoS assurance and load balancing in context-aware service overlay networks," *International Journal of Web and Grid Services*, vol. 11, no. 3, pp. 265-282, January, 2015. Article (CrossRef Link)
[25] P Francois and O Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 6, pp. 1280-1292 , December, 2007. Article (CrossRef Link)
[26] HF Salama, "Multicast routing for real-time communication of high-speed networks," *North Carolina State University*, 1996.

Dan Li received his M.E. and Ph.D. degrees in 2015 and 2018, respectively. He is an assistant research fellow in National Digital Switching System Engineering and Technological R&D Center (NDSC), China. His research interests are future network and routing protocols. Email: pkulidan@foxmail.com

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

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