IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 27, 2018, accepted March 27, 2018, date of publication April 23, 2018, date of current version June 26, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2826925

Optimizing Segment Routing With the Maximum SLD Constraint Using Openflow

LIAORUO HUANG[1], QINGGUO SHEN[1], WENJUAN SHAO [2], AND CUI XIAOYU[1]
[1]Communication Engineering College, PLA Army Engineering University, Nanjing 21007, China
[2]Zijin College, Nanjing University of Science and Technology, Nanjing 21007, China

Corresponding author: Qingguo Shen (shenqg2016@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 61271254.

ABSTRACT Segment routing is an emerging routing technology that was initially driven by commercial vendors to achieve scalable, flexible, and controllable routing. In segment routing, multiple multi-protocol label switch labels are stacked in the packet header to complete end-to-end transmission, which may lead to a large label stack and a long packet header. Thus, scalability issues may occur when segment routing is applied to large-scale networks. To address this issue, multiple mechanisms and algorithms have been proposed for minimizing the label stack size. However, we argue that these methods ignore the constraint on the maximum segment list depth (SLD), since the typical network equipment can currently only support three to five layers of labels. In this paper, we study segment routing with the maximum SLD constraint and demonstrate that issues, such as explosive increases in the size of the label space and the management overheads will arise when the maximum SLD constraint is imposed. To address these issues, we make contributions from two main aspects. First, based on the network programmability that is provided by openflow, a novel segment routing architecture with improved data plane is proposed that reduces the overhead of additional flow entries and label space. Second, a new path encoding scheme is designed to minimize the SLD under the given maximum constraint, while taking multiple types of overhead into consideration. Moreover, we also perform simulations under different scenarios to evaluate the performances of the proposed algorithms. The simulation results demonstrate that the proposed mechanisms and algorithms can address the issues of segment routing when there is a constraint on the maximum SLD.

INDEX TERMS Segment routing, label stack, openflow, path encoding, MPLS.

I. INTRODUCTION

A. BACKGROUND OF SEGMENT ROUTING

Segment routing is a routing technology that can use segments of multiple Label Switch Paths (LSPs) or Interior Gateway Protocol (IGP) routes to complete end-to-end transmission by pushing labels into the packet header at the ingress router. The architecture of the segment routing mainly consists of two parts: a control plane and a data plane. The control plane usually has a centralized controller, which is responsible for routing planning, label advertising, path encoding and so on. The data plane can have a Multi-Protocol Label Switch (MPLS) [1], [2] or IPv6 [3–5]. At present, MPLS is mostly used in the data plane of segment routing. Therefore, the study in this paper is based on the architecture with a data plane with MPLS. In segment routing, the route for each flow is computed by the central controller and mapped into a sequence of segment identifiers (SIDs), which are used to indicate the segments through which the route must pass. At the ingress router, these SIDs are stacked into the packet header in form of MPLS labels. Then, based on the multilayer label forwarding mechanism that is provided by MPLS, packets can utilize multiple segments of various LSPs to complete end-to-end transmission.

The advantages of the segment routing compared with the traditional routing technology, MPLS and source routing can be summarized as follows [1]–[5]:

(1) Flexible: By stacking multiple segment labels in the packet header at the network boundary, segment routing can flexibly control the packet transmission along a specific path or specific nodes according to administrative demand;
(2) Scalable: Because the route information is stored in the labels and determined at the boundary of the network, the per-flow state information is only needed at the

edge routers of the network, and the memory consumption for forwarding entries at the core nodes can be significantly reduced. Thus, the segment routing can be applied in a larger network with more flows;

(3) Service-oriented: In the segment routing, not only the network nodes and links, but also network services such as load balancing and flow filtering can be encoded as SIDs. As a result, packets can be directed to special network function entities, which provides a good support for network function chaining and virtualization [6].

Due to these advantages, segment routing has been supported by increasingly many commercial vendors and is being standardized by IETF. However, although segment routing can bring many benefits to network management, traffic engineering, failure recovery, etc., there are still some issues that need to be addressed. The scalability issue that is caused by label stacking and long packet header is among the most important and fundamental.

B. SEGMENT ROUTING WITH MAXIMUM SLD CONSTRAINT

Segment routing with maximum SLD constraint refers to segment routing under the restriction that the segment list in the packet header can only contain a limited number of SIDs. In segment routing, the longer the end-to-end route, the more labels are needed in the packet header [7]. Therefore, when segment routing is applied to a large-scale network, the number of layers of labels in the packet header may exceed the threshold that the equipment can handle since the typical commercial network equipment can currently only support 3 to 5 layers of labels at most [7]. Then, we argue that there should be a constraint on the maximum segment list depth (SLD) to guarantee the availability of segment routing. However, the paradox is that when there is a constraint on the maximum SLD, there may not be a feasible route from source to destination because it may not be possible to cover all nodes of the route with a limited number of LSPs.

To the best of our knowledge, there is still no relevant research that focuses on this issue. In this paper, we argue that in this situation, an additional segment routing policy [1], including a list of SIDs, should be established to represent multiple segments with a single label. This segment routing policy can be instantiated with a binding SID according to [1] or an LSP that does not follow the shortest-path approach. Since the binding SID requires extra mechanism support such as packet header replacement, in this article, we instantiate the segment routing policy as a new LSP.

To satisfy the constraint of maximum SLD, establishing additional LSPs dynamically will introduce extra overheads such as time consumption and workload and state information expansion in the core routers. In this paper, we are committed to addressing these issues so that the segment routing can both satisfy the maximum SLD constraint and minimize the various overheads.

To illustrate the issues of segment routing with the maximum SLD constraint, we use an example, as shown in Fig. 1.

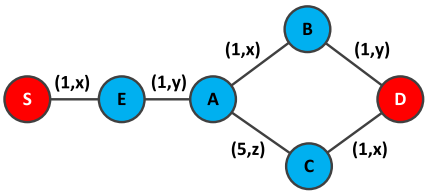


FIGURE 1. Network topology for example.

In Fig. 1, the two-tuple *(n, x)* next to the links consists of the routing metric (*n*) and adjacent SID (*x*) of each link. We suppose that the shortest path to every node has been encoded as a global SID. There is a traffic engineering route S→E→A→C→D from source S to destination D. Then, according to the path encoding algorithms that were introduced in [7–10], the SID sequence with minimum label stack size can be represented as (A, z, D). The label stack depth in the packet header will be 3 in this situation. However, if we assume that the maximum SLD constraint is 2, then an additional segment routing policy (represented as a new LSP) must be established to guarantee that the route has a feasible encoding solution, for example, an LSP along A→C→D. Under the maximum SLD constraint of 2, we can have several solutions, as listed in Table 1. The new segment routing policy is represented as a label P and the average SLD is computed as the sum of the SLD at each hop divided by the number of hops.

TABLE 1. Maximum SLDs of solutions.

Encoding Solution	Additional LSP established	Max SLD	Average SLD	Additional flow entry
(A,z,D)	None	3	2.25	0
(A,P)	P: A→C→D	2	1.5	2
(P,D)	P: S→E→A→C	2	1.75	3
(P)	P:S→E→A→C→D	1	1	4

Based on the data of Table 1, the segment routing may have following issues when there is a maximum SLD constraint:

(1) *Expansion of the forwarding entry number:* In traditional segment routing, the number of forwarding entries in a core router is $|V-1|+|E|$ in the worst case, where V represents the set of nodes and E represents the set of links in the network, because only the shortest paths and the adjacent links must be encoded as SIDs. However, when there is a constraint on the maximum SLD, all possible paths must be encoded. Then, the number of forwarding entries in a core router will expand from $|V-1|+|E|$ to $|(V-1)!|+|E|$ in the worst case. Thus, substantial time and workload need to be spent on establishing and maintaining these LSPs and entries. Meanwhile, many valuable memory resources will be consumed.

(2) *Increasing of the label space:* Because each node, link and network function must be encoded as a label in the segment routing, when the number of additional

segment routing policies and the number of flow entries increase, it leads to the increase of label space consumption, thereby resulting in network performance decline and a shortage of available labels. Although the current label space may be sufficient for segment routing, having fewer labels in the network means that each SID (in the form of a label) can be encoded with fewer bits, thereby resulting in shorter packet headers and fewer flow states being stored in the network core.

(3) *Multiple solutions with the same maximum SLD:* By comparing solution (A, P) and solution (P, D), we observe that solutions with the same maximum SLD may have different overheads of additional established forwarding entries, traffic overload and so on. Therefore, an optimization algorithm is needed to select the best solution from all possible solutions and further reduce the overhead of segment routing.

C. CONTRIBUTIONS OF THIS PAPER

In this paper, we make following main contributions to address the issues that are introduced above:

(1) Segment routing architecture with improved data plane based on Openflow: Although segment routing can reduce the number of routing entries in the core router, it is still possible to require many routing entries under the maximum SLD constraint, as introduced above. We argue that this problem is due in part to the data plane that is used by segment routing being not sufficiently flexible, as each label can only represent a specific LSP to a particular node. In this paper, we use Openflow technology to improve the data plane of segment routing and propose a panel-based forwarding mechanism. The panel consists of node-disjoint LSPs, which can be represented by the same label. Therefore, the label consumption for encoding the SIDs can be reduced. In each panel, we use a label to indicate a particular LSP and a counter to indicate different nodes on this LSP. By this method, the nodes on the same LSP from a specific source can be represented by the same label and various values of the counter without the need for new labels, thereby eliminating the need for additional routing entries.

(2) Formalization of the MaxSLD-PE problem: To minimize the various overheads of segment routing with the maximum SLD constraint, we propose the MAXimum SLD constrained Path Encoding (MaxSLD-PE) problem and formalize the problem as an Integer Linear Programming model. In the optimization model that is proposed in this paper, we not only consider minimizing the maximum SLD but also take into account the flow entry overhead, traffic overhead and routing metric. Furthermore, we argue that by performing the routing algorithm and path encoding independently, we may not able to attain the minimal overhead of segment routing. Then, we combine the path encoding optimization algorithm with the Constrained Shortest Path First algorithm to further optimize the total overhead of segment routing.

(3) Implementation and evaluation: Based on the proposed panel-based forwarding mechanism, we design algorithms for minimizing the label usage in the network. Moreover, we propose algorithms for solving the MaxSLD-PE problem in two application scenarios. Simulations are carried out to evaluate the performance of our solution. According to the simulation results, our algorithm can achieve the minimal overhead comparing with the latest segment routing solutions under the maximum SLD constraint.

The remainder of this article is organized as follows: Section II will survey the related works. The detailed design of segment routing based on Openflow and the panel-based forwarding mechanism are introduced in Section III. In Section IV, we present the formalization and solutions to the MaxSLD-PE problem. We carry out simulations in Section V to evaluate the performance of our solutions. The conclusion is presented in Section VI.

II. RELATED WORKS

There has been a substantial amount of research on segment routing in various application scenarios. In this section, we survey the research that is related to the content of our paper from two aspects.

A. APPLICATION OF SDN IN SEGMENT ROUTING

The combination of Software-Defined Network (SDN) technology with segment routing has been proposed and well-studied, as the SDN features naturally provide good support for segment routing. Some scholars are devoted to putting forward and perfecting the SDN-based segment routing architecture and proving its feasibility through experiments. For example, in Davoli *et al.* [11], propose an architecture of SDN-based segment routing and demonstrate that the full capabilities of MPLS-based segment routing (MPLS-SR) [2] can be implemented with a data plane that consists of SDN-enabled routers. The authors also propose a traffic engineering algorithm that is based on this architecture, which can minimize the average latency of data packets in the network.

Sgambelluri *et al.* [12] propose an SDN-based segment routing architecture for multi-layer packets in an optical network. Through experiments, it is demonstrated that the proposed method can control the label stacking configuration in edge nodes and the dynamic optical bypass can be flexibly performed without requiring the use of signaling protocols.

Moreover, Dugeon *et al.* [13] couple the capabilities of an SDN controller and a path encoding engine to reduce the size of the label stack for expressing segment routing paths.

Other scholars are concerned with the application of SDN-based segment routing in specific scenarios. In Cai *et al.* [14] propose a further evolution of carrier Ethernet architecture by coupling the emerging segment routing and SDN technologies, which can simplify the network infrastructure while providing rich converged services with

embedded high availability and agility. In [15] and [16], the authors devise a segment-routing-based algorithm for optimizing the energy consumption of the backbone network. In addition, Trimponias [17] studied multiple issues that are encountered when using segment routing to implement traffic engineering in a WAN scenario. Lee and Sheu [18] propose an efficient routing algorithm that can meet the user demand while also minimizing the overhead that is brought by the long packet header that is caused by segment routing.

The combinations of SDN and segment routing that are introduced above provide two main benefits: More flexible segment routing can be realized by using the data plane and control plane of SDN. In addition, segment routing can be used to reduce the flow state that is stored in the SDN network, thereby improving the scalability of SDN.

B. PATH ENCODING FOR SEGMENT ROUTING

Because the excessive label stacking may cause scalability issues, optimizing the path encoding for segment routing has become an important problem that needs to be solved. In Giorgetti *et al.* [8], propose algorithms for minimizing the label stack. Lazzeri *et al.* [9] point out that the path encoding solutions with the same maximum SLD may have different overheads due to the different SLDs at each hop. Then, they design an algorithm that computes the best path encoding solutions for equal cost multi paths (ECMPs) between source and destination, while also minimizing the overhead that is caused by label stacking. In addition, Cianfrani *et al.* [10] provide a formulation of and algorithms for solving the path encoding problem, which can obtain the optimal paths for traffic engineering while minimizing the segment list depth.

Although the algorithms that are discussed above can minimize the label stack size by optimizing the path encoding process, they do not consider the maximum SLD constraint. Therefore, it is still possible for the size of the label stack to exceed the value that the device can handle when the end-to-end route is too long. To address this issue, in [9] and [19], the authors propose path encoding optimization algorithms with the maximum SLD constraint. However, in these algorithms, when the label stack size exceeds the maximum SLD constraint, the route is considered unavailable, which may result in there being no available route between the two nodes.

Bidkar *et al.* [20], [21] also studied the path encoding problem of segment routing. They proposed a new data plane technology, which is called Omnipresent Ethernet. Based on this data plane, they researched how to map the SID of the segment routing to the ID of the Omnipresent Ethernet.

C. SUMMARY

In summary, the research that is described above still needs to be further improved in the following aspects:

(1) Although some of the research studies have mentioned the limitations of the current network equipment on maximum SLD, there is still no related research on the issues of segment routing under the maximum SLD constraint. Moreover, based on the research of [9] and [19], currently, the data plane that is used for segment routing is unable to handle the issues of path encoding with the maximum SLD constraint. In addition, the constraint may result in there being no available route and decrease the availability of the segment routing.
To address this issue, it is necessary to improve the data plane using the SDN technique. However, most of the research focuses on how to implement ordinary MPLS-SR using the SDN technique rather than improving it [12]–[23].

(2) The objective of all the proposed path encoding models and algorithms is to minimize the label stack size. Other types of overhead are often regarded as secondary optimization goals or references. As far as we know, none of the models and algorithms have been able to solve the path coding problem under the maximum SLD constraint while taking into account multiple overheads.

Therefore, in the scheme of segment routing with the maximum SLD constraint, various types of overhead should be studied and effective models and algorithms for various issues in this scheme need to be studied. In this paper, to cope with the issues of segment routing with the maximum SLD constraint, we work on two main aspects: improvement of the data plane technology and optimization of the path encoding algorithm.

III. SEGMENT ROUTING BASED ON OPENFLOW

In this section, we mainly focus on improving the data plane of segment routing. First, the architecture of segment routing based on Openflow is introduced. Then, the concept and implementation of the panel-based forwarding are introduced.

A. ARCHITECTURE OF OPENFLOW-BASED SEGMENT ROUTING

The architecture of segment routing based on Openflow is shown in Fig. 2. In this paper, we implement segment routing within the framework of the Software-Defined Network (SDN) [24]. The control plane of segment routing runs in the software controller of OpenDayLight Litmus [25] and the data plane consists of software switches of Open vSwitch 2.3.0 [26]. Through the northern interface, multiple upper applications can be run in the central controller to realize customized network function. To realize the segment routing, we design three software modules: a routing algorithm, segment identifier advertising and a path encoding algorithm.

Initially, every node and link is encoded as a segment identifier. Then, the routes to every SID for each node are computed. These routes are deployed in the form of LSPs by the segment identifier advertising module and corresponding flow entries are installed in the switches. Thereafter, according to the administrative or user requirement, the routing algorithm calculates the best route for each flow based on the network status. Then, the selected route is encoded as

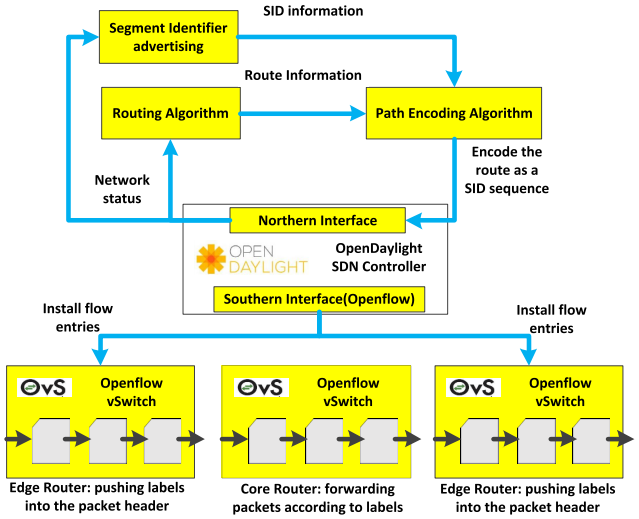


FIGURE 2. Architecture of Openflow-based segment routing.

a sequence of SIDs by the path encoding algorithm. Based on these SID sequences, the flow entries are installed in the edge routers to push labels into the packet header. In the core of the network, each packet is forwarded according to the outermost label, as MPLS does.

Based on the architecture that is proposed above, we have made improvements on the basis of the label-based forwarding protocol that is used by MPLS and propose a panel-based forwarding mechanism for reducing the numbers of labels and flow entries that are consumed by the traditional segment routing data plane.

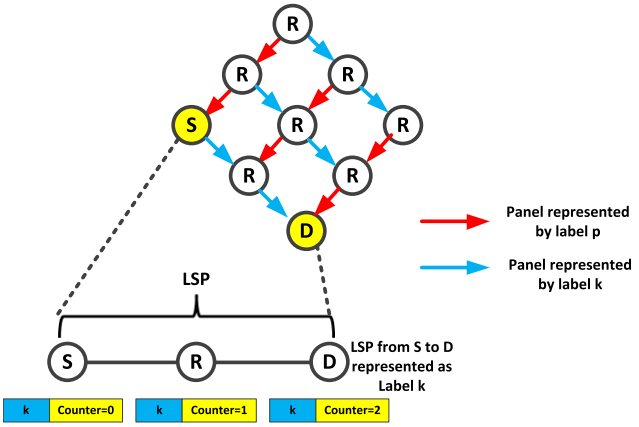


FIGURE 3. Concept of panel-based forwarding.

B. PANEL-BASED FORWARDING

A panel refers to a set of node-disjointed LSPs that can be represented by the same label, as shown in Fig. 3. Since these LSPs are all node-disjointed, packets can still be forwarded unambiguously. Therefore, multiple LSPs can be represented by one label instead of multiple labels, as in ordinary MPLS-SR, and the label space consumption can be reduced. Furthermore, in each panel, we use a label to represent the LSP and use counters to indicate the distance of each node from the source point. Then, each node can be represented as a label with a specific counter, as shown in Fig. 3. In each hop, every time a packet is forwarded according to the outermost label, the counter value in the label is reduced by one.

When the counter value equals 0, the outermost label is removed. Since the operation of the counter is very similar to that of the Time To Live (TTL) and Openflow fully supports the operations on the TTL, in this paper, we encode the counter value into the TTL field of the label. Moreover, we set a threshold for the TTL as the 0 value of the counter. Thus, the counter value equals the TTL value minus the threshold and the operations of the TTL and the counter can be made fully compatible.

Based on the panel-based forwarding mechanism that is introduced above, both global SID and adjacent SID can be represented as labels with specific counters. Accordingly, the encoded sequence of the end-to-end route also changes to a stack of labels with counters. In this paper, we implement panel-based forwarding with Openflow technology. Six main procedures need to be performed:

(1) LSP planning: In this procedure, the routes to every node are all computed by the central controller according to the shortest-path-first algorithm. Then, these routes are presented as LSPs and denoted by SIDs.

(2) LSP labeling: According to the panel-based forwarding mechanism that is introduced above, in this procedure, we divide the LSPs into as few panels as possible to minimize the label usage. To obtain the minimal label consumption, we use an optimization model of LSP labeling, which is presented in Section IV. The LSP labeling procedure can be performed in the controller as a standalone application or as part of the segment identifier advertising module.

(3) Route computing: In this procedure, we compute the best route for each incoming flow using routing algorithms such as widest path first, constrained shortest path first, and the minimum-interference routing algorithm.

(4) Path encoding: Path encoding is the process of mapping the route to a sequence of SIDs. According to these SIDs, the corresponding segment labels that must be pushed at the ingress router can be determined. However, since there may be multiple solutions to encoding a given route, different encoding strategies may lead to different label stack sizes in the packet header and other overheads such as traffic overload and forwarding entry consumption. Thus, the performance of the path encoding algorithm will be directly related to the scalability of segment routing. Therefore, we design mathematical models and algorithms for optimizing this process in Section IV.

(5) Packet programming: In this procedure, labels are pushed into the packet header according to the preinstalled flow entries at the edge routers. Since Openflow can fully support the TTL operations, in this paper,

we exploit the MPLS label format and use its TTL field as the counter. Based on the operations that are provided by Openflow, the flow entry structure and processing logic in the edge router are shown in Fig. 4. When a new packet arrives, labels are pushed layer by layer using the action of Push MPLS Label, and the counter value of each label is also set using the action of Set MPLS TTL according to the pre-computed end-to-end route.

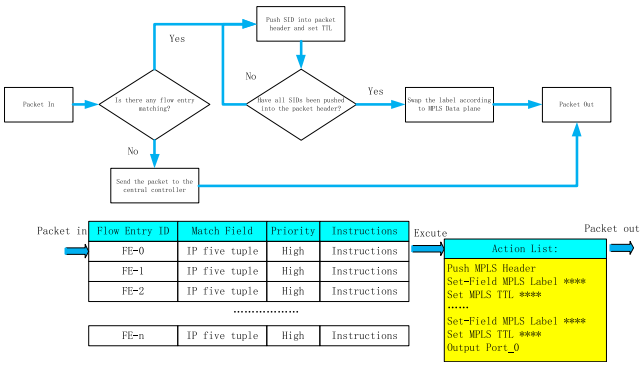


FIGURE 4. Packet programming procedure.

(6) *Packet forwarding:* This procedure is performed in the core routers to forward packets according to the label stack in the packet header. As we introduced earlier, each time a packet is forwarded according to the outermost label, the TTL value of the label is reduced by one. When the TTL value of the outermost label equals the threshold, the outermost label is popped and the packet continues to be forwarded according to the inner label. The flow entry structure and processing logic of packet forwarding are shown in Fig. 5.

C. EXAMPLE

We use an example to illustrate the working principles of panel-based forwarding. Consider the MPLS network that is shown in Fig. 6. There are three LSPs: S→R2, R1→R3→D, and R2→R3→R4. These LSPs are classified into two panels, which are represented by blue and red in Fig. 6. The red panel is labeled as B and the blue one is labeled as A. We assume that the end-to-end route is S→R2→R3→D. Then, three layers of labels are stacked in the packet header at the ingress router S. The packet is first transmitted along LSP S→R2. At router R2, the counter of the outermost label is reduced to 0. Then, the outermost label is removed and the packet continues to be transmitted to R3 along LSP A. After the same operations are performed in router R3, the packet is delivered to the destination D. In this case, with panel-based forwarding, only two labels are needed in the network and no new LSP or flow entries must be created. However, with the ordinary MPLS-SR, a new LSP R2→R3 must be established and the number of required labels is increased to 4.

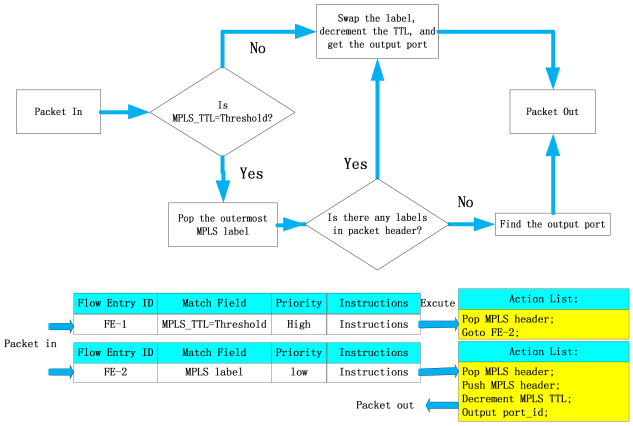


FIGURE 5. Packet forwarding procedure.

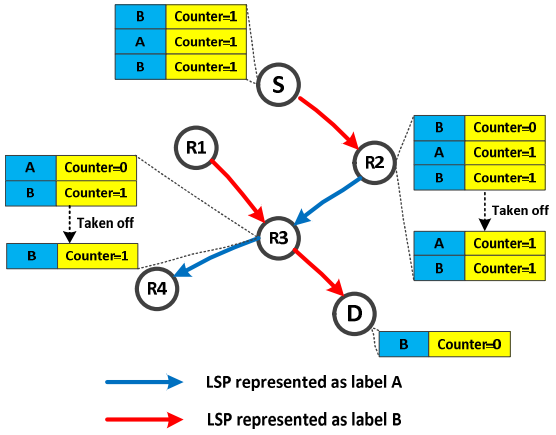


FIGURE 6. Network topology for example.

D. DISCUSSION

According to the above example, the panel-based forwarding mechanism that is proposed in this paper can be considered an improved version of the traditional label-based forwarding mechanism. Comparing with the traditional data plane of segment routing, the improved data plane that is proposed in this paper has the following main advantages:

(1) Less label usage: Since multiple LSPs can be represented by the same label and the nodes in an LSP can be represented by the same label with different counter values, the label usage in the network can be significantly reduced.

(2) Fewer required forwarding flow entries: Since different nodes that belong to the same LSP can be represented by the same label with different counter values, the number of forwarding flow entries that are required in the network can be reduced. In Section V, we will experimentally demonstrate the effectiveness of the proposed panel-based forwarding mechanism.

(3) More flexible route planning: In traditional label based forwarding, a packet can enter the tunnel at any point, but can only leave the tunnel at the end. However, in

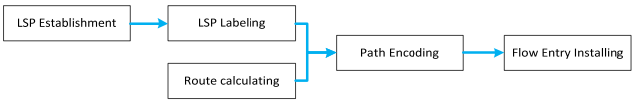


FIGURE 7. Workflow of the segment routing based on Openflow.

TABLE 2. Notations used in this paper.

Symbol	Definition
e_{ij}^{f}	equals 1 if flow f passes through link (i,j);
ε_{ij}^{p}	equals 1 if the LSP p passes through link (i,j);
$\eta_{ij}^{f,p}$	equals 1 if the flow f is carried by LSP p at link (i,j);
d_{ij}^{f}	the SLD of flow f at link (i,j)
D_i^f	the layers of labels that are required to transmit flow f from source to node i;
m_{ij}	the routing metric of the link (i,j)
r_f	the route for the flow f;
k_{p1}^{c}	equals 1 if the LSP p_1 is represented by label c;
Ψ_f	the flow entry overhead of flow f;
Γ_f	the traffic overhead of flow f;
Λ_f	the maximum SLD of flow f;
N	the maximum SLD constraint

panel-based forwarding, the packet can leave the tunnel at any point through setting the appropriate TTL value in the label. Therefore, the packet can use any segment of a label switch path to complete the end-to-end transmission rather than only using the entire path, like MPLS does.

IV. FORMALIZATION OF THE OPTIMIZATION PROBLEMS

Based on the proposed panel-based forwarding mechanism, the workflow of the segment routing based on Openflow can be summarized as follows:

To reduce the overhead of segment routing under the maximum SLD constraint, we need to optimize two of the processes: LSP labeling and path encoding. The objective of the LSP labeling optimization is to reduce the label usage in the network and the number of new flow entries that need to be installed under the maximum SLD constraint.

Based on the result of LSP labeling optimization, path encoding optimization can be performed. The objective of the path encoding process is to minimize the label stack size in the packet header and the multiple types of overheads. In this section, we introduce the optimization models of these two process and design algorithms for solving them.

Furthermore, we combine the path encoding algorithm with a routing algorithm. Taking into account the overhead of the path encoding algorithm that is based on panel-based forwarding, the route of each flow can be further optimized to achieve the minimal overhead of segment routing. The notations that are used in this section are listed in Tab. 2. The network is denoted as a bidirectional graph G(E, V), where E is the set of edges and V is the set of vertices.

A. LSP LABELING OPTIMIZATION

The objective of the LSP labeling optimization model is to minimize the label usage while ensuring that the LSPs that are represented by the same label are all node-disjointed. We use vector $K^c = \{k_{p1}^c, k_{p2}^c, \ldots\ldots k_{pn}^c\}$ to denote the LSPs that are represented by label c. We present Theorem 1 as follows:

Theorem 1: If two LSPs, which are denoted as p_1 and p_2, satisfy formula (1) for node n

$$\sum_{m} \varepsilon_{nm}^{p_1} + \sum_{q} \varepsilon_{nq}^{p_2} \leq 1 \tag{1}$$

then node n can unambiguously forward packets of p_1 or p_2 if they use the same label.

Proof: It is easy to prove that $\sum_{m} \varepsilon_{nm}^{p_1} = 1$, $\sum_{q} \varepsilon_{nq}^{p_2} = 1$ is the necessary and sufficient condition for p_1 and p_2 to both pass through node n. If p_1 and p_2 are represented by the same label, then the label will be ambiguous at node n. Therefore, p_1 and p_2 cannot pass through n at the same time. Theorem 1 is proved.

We assume that the number of labels c is equal to the number of LSPs and the set of c is represented as C, thereby ensuring that each LSP can be represented by at least one label. Then, we define the maximum value of k_p^c in vector K^c as μ^c such that μ^c equals 0 or 1. Thus, μ^c can be used to indicate whether the label c is used to represent an LSP, and $\sum_{c} \mu^c$ can be used to denote the label usage. Then, the objective function of the optimization model can be expressed as:

$$\text{minimize} \sum_{c} \mu^c \tag{2}$$

According to the definition of μ^c, we define the constraints as follows:

$$k_{p_i}^c \leq \mu^c, \quad \forall p_i \in P, \ \forall c \in C \tag{3}$$

To guarantee that LSPs that are represented by the same label are all node-disjoint, we define a constraint by formula (4) according to Theorem 1:

$$k_{p_i}^c \sum_{m} \varepsilon_{nm}^{p_i} + k_{p_j}^c \sum_{q} \varepsilon_{nq}^{p_j} \leq 1, \quad \forall (n,m) \in E, \ (q,n) \in E \tag{4}$$

According to the above analysis, the LSP labeling problem can be viewed as a classic vertex coloring problem. In this paper, we represent each LSP by a vertex in a virtual graph. When two LSPs have the same node, the two vertices that represent these two LSPs are connected by an edge. Then, the LSP labeling problem is transformed into a vertex coloring problem, which means we need to color vertices in the virtual graph with as few colors as possible and ensure that no adjacent vertices are of the same color. The back-tracing method is used to solve this problem and the pseudocode of the algorithm is shown as Algorithm 1.

As Algorithm 1 shows, we first construct the virtual graph of LSPs as $G'(V', E')$, in which each vertex represents an

Algorithm 1 LSPLableing

input: network topology G;
output: labeling solution $\{K\}_c$;
construct the virtual graph $G'(V', E')$;
for $m \leftarrow 0$ **to** $|V'|$ **do**
  $feasible \leftarrow$false;
  for $i \leftarrow 0$ **to** $|V'|$ **do**$C[i] \leftarrow 0$;
  $v \leftarrow 0$;
  while $v \geq 0$ **do**
    $C[v] \leftarrow C[v]+1$;
    while $C[v]<m$ **do**
      if the coloring solution is feasible **then**
      break;
        else then $C[k] \leftarrow C[v] + 1$;
        end if;
        end while
        if $C[v] \leq m$ and $v = |V'| - 1$ **then**
          $feasible \leftarrow$true; **break;**
        else if $C[v]>m$ and $v = 0$ **then**
          $feasible \leftarrow$false; **break;**
        else if $C[v] \leq m$ and $v< |V'|-1$ **then**
          $v \leftarrow v+1$;
        else if $C[v]>m$ and $v>0$ **then**
          $v \leftarrow v-1$;
        end if;
      end while;
      if $feasible =$ true **then**
        convert the vector C to the labeling solution $\{K_c\}$.
        break;
      end if;
    end for;

LSP and each edge indicates whether two LSPs are node-disjoint. Vector $C[n]$ stores the coloring scheme for the vertices and the maximal number of colors is m. In each iteration, we set $C[v]$ as $C[v]+1$. If $C[v]$ has no conflict with any other neighboring vertex, we then continue coloring the next vertex; otherwise, we back-trace and test the next color. The Boolean variable *feasible* indicates whether the vertices of the virtual graph can be colored with m colors. If *feasible* is true, then we find the optimal solution with the minimal number of colors and covert the coloring scheme C to the labeling solution K. The complexity of constructing the virtual graph is $O(|V'|(|V'|-1)/2) = O(|V'|^2/2)$. The complexity of coloring the graph with m colors is $O(m^{|V'|})$. Thus, the total complexity of the algorithm is $O(\sum_{m=1}^{|V'|-1} m^{|V'|} + |V'|^2/2)$ in the worst case. Because the complexity of Algorithm 1 is too high for it to be applied to complex or large-scale networks, a simplified algorithm is shown as Algorithm 2.

As Algorithm 2 shows, for each node in the graph, we try to use the colors that have already been used to minimize the total number of colors that are used. The complexity of

Algorithm 2 LSPLableing

input: network topology G; the set of LSP;
output: labeling solution $\{K_c\}$;
construct the virtual graph;
for $i \leftarrow 0$ **to** the total number of LSP **do**
for $j \leftarrow 0$ **to** $i+1$ **do**
  if coloring the ith node with the jth color is feasible **then**;
  color the ith node with the jth color;
  break;
  end if;
end for;
end for;

Algorithm 2 is $O(|V'|^2/2 + |V'|^2)$ as the complexity of each iteration is $O(|V'|^2)$. Although Algorithm 2 can only obtain an approximately optimal solution, we will prove in Section V that Algorithm 2 can achieve satisfactory label reduction performance with very low complexity in most scenarios.

After Algorithm 2 is performed, panels can be established with suitable colors. When a new LSP needs to be set up, we will examine all the labels of the existing panels for representing the new LSP. When there is a label that can guarantee that the new LSP is node-disjointed with other LSPs in the same panel, the label is feasible; otherwise, we use a new label to represent the new LSP.

B. PATH ENCODING OPTIMIZATION

Based on the result of the LSP labeling process, path encoding optimization can be performed. Before introducing the optimization model, we first introduce the overheads that are used in this subsection as follows.

In practice, a larger SLD may lead to a more complex processing mechanism of the device and a longer packet header. Thus, in previous research, the authors focused on minimizing the SLD of the packet header. We can define the maximum SLD as follows:

Definition 1: The maximum segment list depth for the flow f can be expressed as:

$$\Lambda_f = \max\{d_{ij}^f \bullet e_{ij}^f\} \tag{5}$$

According to the panel-based forwarding mechanism that was introduced in Section III, when a route is carried by different LSPs on two adjacent links, then two labels are used in the packet header for these two LSPs. Therefore, we can rewrite the maximum SLD as follows:

$$\Lambda_f = \sum_j \left[\frac{1}{2} \sum_p \left| \eta_{kj}^{f,p} - \eta_{jv}^{f,p} \right| \right] \tag{6}$$

Furthermore, the traffic overhead that is caused by stacking multiple labels should be considered. In [5], the authors point out that traffic overhead is related not only to the maximum SLD but also to the SLD at each hop. Therefore, path

encoding schemes with the same maximum SLD may have different traffic overheads. We define the traffic overhead as the sum of the SLD at every hop as follows:

Definition 2: The traffic overhead of the flow f can be expressed as:

$$\Gamma_f = \sum_{(i,j)} d_{ij}^f \tag{7}$$

In addition to the two types of overhead that are introduced above, as introduced in the previous section, under the maximum SLD constraint, new flow entries may need to be installed. Since installing flow entries may cause extra overhead of workload and time, we should also consider the flow entry overhead. In this paper, we suppose that there is a virtual LSP $\tilde{p}$ that covers every link through which the route passes. If a segment of the route can only be carried by this virtual LSP, then in each node of the segment, no existing flow entries can be used for the route and new flow entries must be installed. Therefore, the maximum flow entry overhead equals the number of links that are carried by the virtual LSP. We define the flow entry overhead as follows:

Definition 3: The flow entry overhead of the flow f can be expressed as:

$$\Psi_f = \sum_{(i,j)} n_{ij}^{f,\tilde{p}} \tag{8}$$

The maximum value of Ψ_f is no larger than the length of the route. In the path encoding process, the three types of overhead should all be taken into account and the path encoding algorithm should be optimized to minimize the overhead that is defined above. In this paper, since the maximum SLD is constrained, we introduce parameter α to indicate the importance of the traffic overhead and parameter β to indicate the importance of the flow entry overhead, and set $\alpha + \beta = 1$. Then, based on the analysis of [30] and [31], we can express the objective function of the optimization model as follows:

$$\text{minimize}\, \alpha \sum_{(i,j)} d_{ij}^f + \beta \sum_{(i,j)} \eta_{ij}^{f,\tilde{p}} \tag{9}$$

For applications in which the bandwidth is relatively tight, a larger value of α should be selected. However, when larger processing overhead occurs for installing new flow entries or new LSPs, a larger value of β should be set. The consistency constraints can be expressed as:

$$\sum_{p} \eta_{ij}^{f,p} = e_{ij}^f, \quad \forall p \in P,\ \forall f \in F \tag{10}$$

$$\eta_{ij}^{f,p} \le \varepsilon_{ij}^p, \quad \forall p \in P,\ \forall f \in F \tag{11}$$

Formula (10) ensures that the flow is transmitted by an LSP at the ith link. Formula (11) means that the path must pass the link where it carries the traffic. The loop-free and continuity constraints for $\tilde{p}$ can be expressed as:

$$\sum_{i} \varepsilon_{ij}^{\tilde{p}} \le 1, \quad \forall i \in V,\ \forall (i,j) \in E \tag{12}$$

$$\sum_{k} \varepsilon_{kj}^{\tilde{p}} - \sum_{v} \varepsilon_{jv}^{\tilde{p}} = \begin{cases} -1j \text{ is the source node} \\ 0j \text{ is the intermediate node} \\ 1j \text{ is the destination node,} \end{cases} \quad \forall k, v, j \in V \tag{13}$$

Next, we obtain the maximum segment list depth constraint according to formula (14) as:

$$\sum_{j} \left[\frac{1}{2} \sum_{p} \left| \eta_{kj}^{f,p} - \eta_{jv}^{f,p} \right| \right] \le N, \quad \forall e_{kj}^f = 1,\ \forall e_{jv}^f = 1,\ \forall p \in P \tag{14}$$

This mathematical model defines an Integer Linear Programming problem. Next, we will introduce an algorithm for solving this problem. When the maximum SLD is N, the route can be broken into at most N segments. Thus, there are *N-1* break points at most. In the algorithm, we first construct a virtual LSP that covers every link through which the route passes. Then, a recursive function is designed, which can install *N-1* break points for traversing all nodes of the route. Therefore, all routing segmentation schemes can be traversed. For each segmentation scheme, we calculate their cost values and select the minimum one as the best solution.

The length of the end-to-end route is expressed as $\sum_{(i,j)} e_{ij}^f$ and is $|V|$ in the worst case. Therefore, the complexity of finding all possible segmentations is $O(|V|!/[(|V| - N + 1)!(N + 1)!])$ in the worst case. However, since generally the length of an end-to-end route is less than 20 and N is less than 10, Algorithm 3 is not time-consuming in practical applications.

C. ROUTE PLANNING OPTIMIZATION

Traditionally, the route planning process and path encoding process are conducted independently. However, since the optimization goal of route planning is usually different from that of path encoding, the optimal result may not be obtained when the two processes are executed separately. To address this issue, we combine the route planning process with the path encoding process. In addition to the cost of path encoding, we also considered the cost of the routing algorithm in the optimization model. The objective function of the optimization model can be represented as:

$$\text{minimize}\, \gamma_1 \sum_{(i,j)} e_{ij}^f m_{ij} + \gamma_2 \Psi_f + \gamma_3 \Gamma_f \tag{15}$$

where γ_1, γ_2, and γ_3 are the adjustment parameters of each metric. In addition to the constraints in formulas (10)∼(13), the end-to-end route for the traffic should also be loop-free and continuous, as shown in (16)∼(17).

$$\sum_{i} e_{ij}^f \le 1, \quad \forall i \in V,\ \forall (i,j) \in E \tag{16}$$

$$\sum_k e^f_{kj} - \sum_v e^f_{jv} = \begin{cases} -1j \text{ is the source node} \\ 0j \text{ is the intermediate node} \\ 1j \text{ is the destination node,} \end{cases} \quad \forall k, v, j \in V \quad (17)$$

The maximum label depth constraint can be represented as:

$$\sum_j \left[\frac{e^f_{kj} e^f_{jv}}{2} \sum_p \left| \eta^{f,p}_{kj} - \eta^{f,p}_{jv} \right| \right] \le N, \quad (18)$$

$$\forall (k, j) \in E, (j, v) \in E, \forall j \in V, \forall p \in P \quad (19)$$

We design an algorithm for solving the Integer Programming problem that is defined above. First, we use the CSPF algorithm to find the optimal route. Then, we calculate the cost of path encoding for this route according to Algorithm 3 and use this cost as the baseline $cost_{baseline}$ *(s, d)* to examine every possible path. To obtain the best solution, we must find all possible routes from the source to the destination. However, this process may be very time-consuming since the complexity is very high when applied in a large-scale network. To reduce the time consumption of the algorithm, we need to exclude routes that are identified as not optimal at an early stage. We obtain a judgment rule, which is defined in Theorem 2:

Algorithm 3 OptimalBreakpoint

input: N;$\{e^f_{ij}\}$; $\{\varepsilon^f_{ij}\}$;
output: $\{\eta^f_{ij}\}$;
construct the virtual LSP $\tilde{p}$;
construct a list *points* and put the first node of the route into *points*;
breakpoint(0, N-1, points);

Function breakpoint(i, depth, points)
depth–
for $i \leftarrow$ *start* **to** length of the route **do**
put the *i*th node of the route into *points*;
if *depth*>0 **then**
breakpoint(i, depth, points);
remove the last node from *points*;
else then
put the last node of the route into *points*;
compute the cost value of solution *points*;
if *points* has the minimum cost **then**
make *points* the optimal solution;
end if
end if
end for

Theorem 2: For any segment from node k to node p of route r, if

$$\gamma_1 \sum_{(i,j)} e^f_{ij} m_{ij} + \gamma_3 \sum_{(i,j)} e^f_{ij} \ge cost_{baseline}(s, d), \quad \forall (i, j) \in segment \quad (20)$$

then route r is not optimal.

Proof: According to Definition 1 to Definition 3, the minimal values of Ψ_f and Γ_f are 0 and $\sum_{(i,j)} e^f_{ij}$, respectively. Therefore, the minimal total cost of segment *(k, p)* is $\gamma_1 \sum_{(i,j)} e^f_{ij} m_{ij} + \gamma_3 \sum_{(i,j)} e^f_{ij}$. Thus, when the minimal cost of a segment in route r is larger than the baseline, route r is not more optimal than the shortest path.

Algorithm 4 Routing Combined With Path Encoding

input: N; $\{\varepsilon^f_{ij}\}$;
output: $\{\eta^f_{ij}\}$; $\{e^f_{ij}\}$;
find the constrained shortest path using the CSPF algorithm;
compute the overhead of r;
create the initial solution r_0 and put the source node into r_0;
put r_0 into the possible route set R;
$i \leftarrow 2$;
while $i < |V|$ **do**
while the length of first route r_0 in set R is less than i **do**
find the last node of r_0;
if the last node of r_0 equals the destination **then**
put r_0 into the solution set S;
else then
for $k \leftarrow 0$ **to** $|V|$ **do**
if node k is connected with the last node of r_0 and node k is not contained in r_0 **then**
create the same route r_{temp} as r_0;
put node k into r_{temp};
if $cost(r_{temp}) < cost(r)$ **then**
put r_{temp} into the set R according to Theorem 2;
end if;
end if;
end for;
end if;
remove r_0 from R;
end while;
$i++$;
end while;
for every route in set S **do**
apply Algorithm 3 to each route;
end for;
select the optimal route with minimal cost;

The algorithm that we utilize is presented as Algorithm 4. First, we calculate the shortest path using CSPF. Then, we set the total cost of the shortest path as the baseline. Thereafter, all possible routes from source to destination are calculated

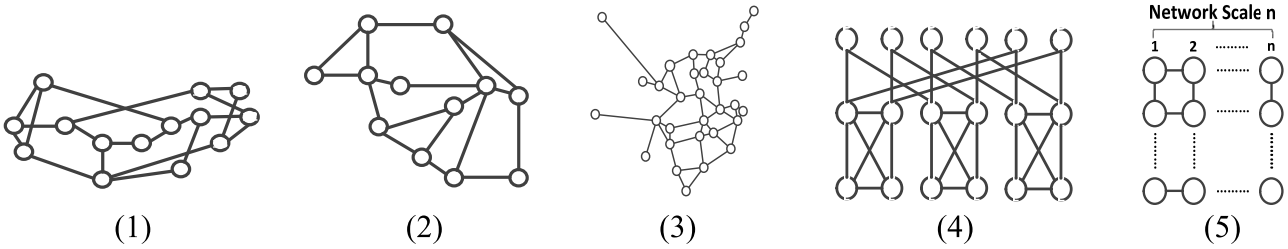


FIGURE 8. Test topologies for simulation. (a) NSFNet. (b) Polska. (c) ChinaNet. (d) FatTree. (e) Matrix.

with Widest-First Search. In each iteration, according to Theorem 2, most non-optimal routes are deleted. Last, to every remaining possible route, we apply Algorithm 3 and select the optimal route with minimal cost.

The complexity of finding the shortest path with CSPF is $O(|V|log|E|)$. The complexity of finding all possible solutions in the worst case is $O((|V|-1)!)$. The complexity of applying Algorithm 3 to every remaining route is $O((|V|-1)!|V|!/[(|V|-N+1)!(N+1)!])$. Then, the total complexity of Algorithm 3 is $O(|V|log|E|+(|V|-1)!+(|V|-1)!|V|!/[(|V|-N+1)!(N+1)!])$. In practice, most non-optimal solutions can be eliminated according to Theorem 2 in the early stage of the algorithm. Therefore, Algorithm 3 is not time-consuming in most scenarios.

V. EVALUATIONS

A. SIMULATION SETTINGS

In this section, we will evaluate the effectiveness of the proposed mechanisms and explore the factors that influence the performances of the algorithms under different application scenarios. In simulations, we use five types of topologies, as shown in Fig. 8: NSFNet, Polska, ChinaNet, FatTree, and Matrix. The former four topologies are typical backbone networks that are used to evaluate the effectiveness of the proposed method in the practical application scenarios. The Matrix topology is a generalized topology that is used to study the factors that affect the performances of the algorithms. The network size of the Matrix topology can be adjusted by setting the scale parameter n. When the scale of the Matrix network is n, the number of nodes in the network is n×n. Each link of the topology has two parameters: metric and available bandwidth. Metric is an additive indicator that can be used to simulate latency and administrative metric. Available bandwidth is a concavity indicator, which can also be used to simulate the transmission rate. The metric of each link is randomly distributed in [0, 5] and the available bandwidth is randomly distributed in [0, 100M].

Moreover, to evaluate the performance of the path encoding algorithm under different routing algorithms, we use three typical routing algorithms in this paper: Minimum-Interference Routing algorithm (MIRA), Widest-Path-First algorithm (WPF), Constrained Shortest-Path-First algorithm (CSPF).

(1) Minimum-Interference Routing Algorithm (MIRA) [27]: The purpose of this routing algorithm is to choose the route with the least impact on the traffic of other links between the source and destination nodes. By calculating the weights for every link, the Dijkstra algorithm is used to minimize the interference.

(2) Widest-Path-First Algorithm (WPF) [28]: This algorithm is a modification of the Dijkstra algorithm. It takes the largest metric of all links in the path as the metric of the path. Then, the path from source to destination with the largest metric is selected.

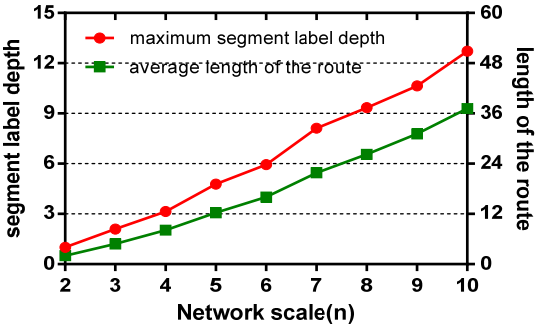
(3) Constrained Shortest-Path-First Algorithm (CSPF) [29]: This routing algorithm is usually used in traffic engineering. CSPF is the Shortest-Path-First routing with multiple constraints. In this paper, we are mainly concerned with the constraint on the available bandwidth.

As the initial configuration of the simulation, LSPs are established according to the Shortest-Path-First algorithm, which ensures that any two points in the network are connected by at least one LSP. In practice, ISPs focus more on the overhead of installing new flow entries than on the traffic overhead that is caused by extra layers of labels in the packet header. Therefore, we set α as 0.8 and β as 0.2. In the route planning scheme, $\gamma 1$, $\gamma 2$ and $\gamma 3$ are set to 0.5, 0.4 and 0.1, respectively. However, determining the best parameter value is beyond the scope of this paper and will be studied in the next step of our research.

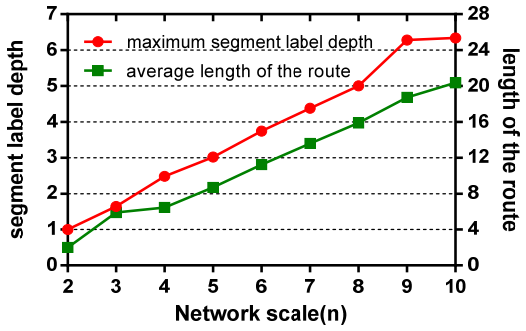
To demonstrate the superiority of the proposed mechanisms and algorithms, we consider the following state-of-the-art encoding algorithms as comparison methods:

(1) Minimum Max SLD path encoding without Maximum SLD constraint (minSLD-without Constraint): This is the main path encoding strategy that is used for segment routing. The objective of this algorithm is to find a path encoding solution with minimum max SLD so that the size of the label stack in the packet header can be minimized. Moreover, according to [5], we also consider the traffic overhead that is caused by different path encoding strategies.

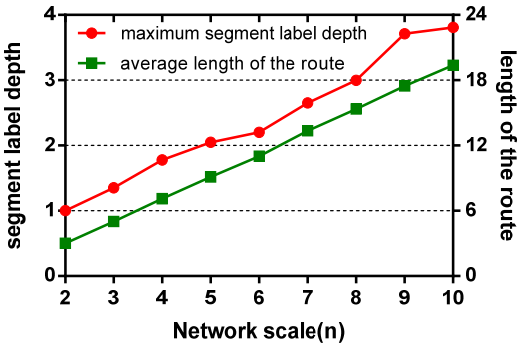
(2) Minimum number of new flow entry path encoding with Maximum SLD constraint (minEntry-Max SLD Constraint): As we introduced in Sections I∼III, when there is a maximum SLD constraint for the path encoding, new flow entries may have to be installed. Since installing new flow entries may be time-consuming, the purpose of this algorithm is to find the path



(1)



(2)



(3)

FIGURE 9. Maximum SLDs with different network sizes. (a) Path encoding with the MIRA routing algorithm. (b) Path encoding with the WPF routing algorithm. (c) Path encoding with the CSPF routing algorithm.

encoding solution with the minimum number of new flow entries to be installed.

In addition to the total cost, which is defined in Section IV, we evaluate the average SLD, the maximum SLD and the number of additional established flow entries to examine the performances of different methods.

B. DEMONSTRATION OF THE MAXSLD-PE PROBLEM

The simulation results of max SLD with expanding network size in the Matrix network are shown in Fig. 9.

According to Fig. 9, the length of the end-to-end route increases with the size of the network. At the same time, regardless of which routing algorithm is used, the maximum SLD that is required for path encoding always grows. Since the LSPs are established according to shortest-path-first algorithm, the more closely the calculated route conforms to the shortest path, the lower the maximum SLD that is needed in the path encoding process. Thus, the routes that are calculated by the MIRA and WPF algorithms require larger values of SLD to complete end-to-end transmission comparing with that calculated by CSPF. Moreover, in the worst case, the maximum SLD grows to more than 12 for the MIRA algorithm, which far exceeds the depth of label stacks that the current device can support. Therefore, considering the actual processing ability of the current network devices, the maximum SLD constraint should be set during the path encoding process.

C. IMPACT OF THE MAXIMUM SLD CONSTRAINT

According to the analysis in Section III, it may be necessary to install new flow entries to satisfy the maximum SLD constraint. Because installing new flow entries introduces extra overhead and workload, the ISP needs to know the impact of the maximum SLD value on the number of new flow entries to be installed, so that the appropriate constraint value can be selected in actual deployment. In this section, we conduct an evaluation in a matrix network under three routing algorithms. The source and the destination of the route are randomly selected. The simulation is conducted 50 times and the results are averaged, as shown in Fig. 10.

According to the simulation results, as the network size grows, the number of new flow entries to be installed increases to satisfy the constraint of maximum SLD, regardless of which routing algorithm is used. Meanwhile, the larger the constraint value is, the fewer flow entries need to be installed. The reason for this phenomenon is that a larger SLD constraint means that the packet can utilize more segments to accomplish the end-to-end transmission without the need to create a new LSP. Similar to the situation that is considered in Section V.B, the more closely the calculated route conforms to the shortest path, the fewer flow entries need to be installed to fulfill the maximum SLD constraint.

For practical applications, the value of the maximum SLD constraint should be carefully selected. If the constraint value is too large, the device complexity and processing cost will increase. In contrast, if the constraint value is too small, there will be too many new flow entries to be installed. However, choosing the most suitable maximum SLD constraint value is beyond the scope of this paper. In the following simulations, we simply set the constraint value to 3.

D. EVALUATION OF THE IMPROVED DATA PLANE

As introduced in Section III, we have made improvements to the ordinary MPLS-SR data plane that is used in segment routing to further reduce the number of flow entries and label usage. In this section, we use simulations to evaluate the effectiveness of the proposed panel-based forwarding mechanism. We assume that the LSPs are all pre-established by the Shortest-Path-First algorithm. Each experiment is conducted 50 times and the results are averaged.

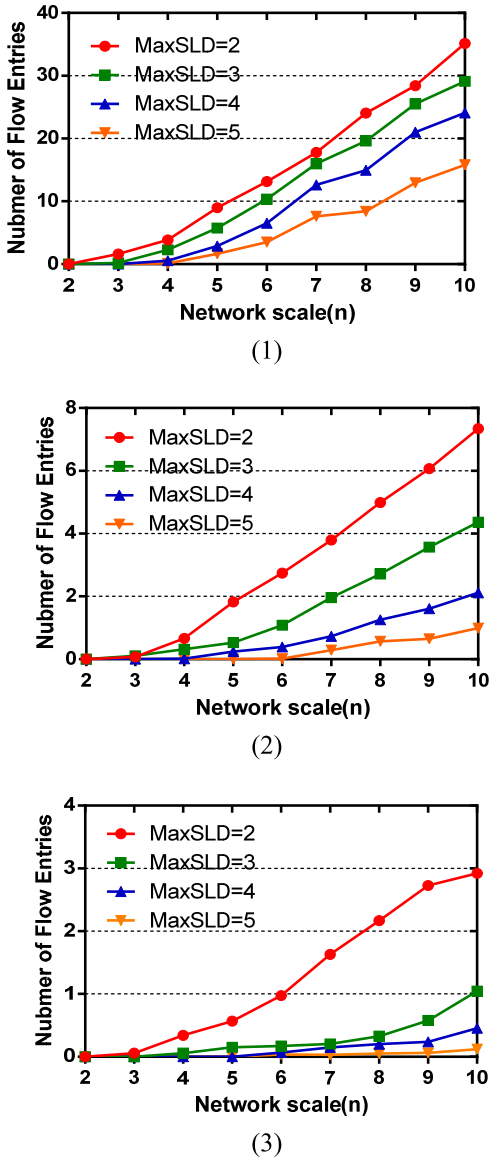


FIGURE 10. Numbers of net flow entries with different maximum SLD constraints. (a) Path encoding with the MIRA routing algorithm. (b) Path encoding with the WPF routing algorithm. (c) Path encoding with the CSPF routing algorithm.

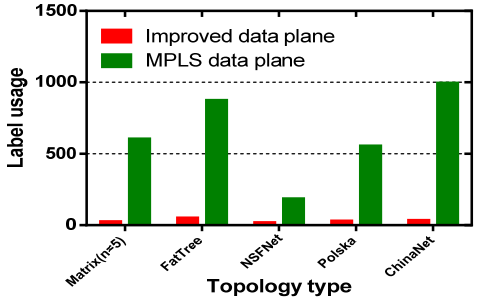


FIGURE 11. Label usage reduction in test topologies.

First, we verify the effectiveness of the improved data plane in reducing label usage under different topologies. The result is shown in Fig. 11.

Although the panel-based forwarding method can achieve different label reduction effects under different topologies, it can substantially reduce the number of required labels in the network regardless of the topology. Under the ChinaNet topology, it achieves the best results with a reduction of 96.8% of the label usage. Even in the worst-case scenario (NSFNet), it reduces label usage by approximately 91.2%. Therefore, we conclude that the proposed panel-based forwarding method can achieve a significant label reduction effect under a variety of topologies and can be applied to real backbone networks.

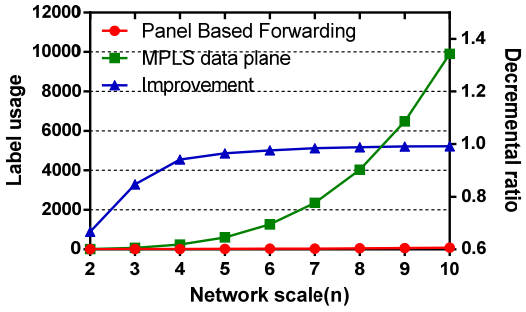


FIGURE 12. Performance of the proposed method under different network scales.

In Fig. 12-14, simulations are conducted for a Matrix network. The performance of the proposed panel-based forwarding method varies with the size of the network. As network scale increases, the advantages of the proposed panel-based forwarding method over MPLS continue to increase and the best results are achieved when n reaches 10, in which the label usage is reduced by approximately 99.2% (as Fig. 12 shows). We conclude that the proposed method can be applied in large-scale networks and its advantage increases with the network size.

Moreover, we evaluate the impact of the proposed method on the number of additional flow entries that are required for satisfying the maximum SLD constraint. Based on the path encoding algorithm of Algorithm 3, under different routing algorithms, the numbers of additional flow entries that need to be installed with the proposed mechanism and ordinary MPLS-SR are shown in Fig. 13.

The simulation results show that although the number of new flow entries that need to be installed increases with the network size, the number of new flow entries can always be reduced by using the improved data plane. Moreover, the advantage that is achieved by using the improved data plane also increases with the network scale. According to the result in Fig. 13, with the proposed panel-based forwarding mechanism, the number of new flow entries that need to be installed is also reduced. In the best case (CSPF routing algorithm), the required number of flow entries is reduced by 86.4%. Even in the worst case, it reduces the number of flow entries by 25.6%. Due to the decrease of the number of flow entries, the total cost of path encoding can also be reduced, as shown in Fig. 14.

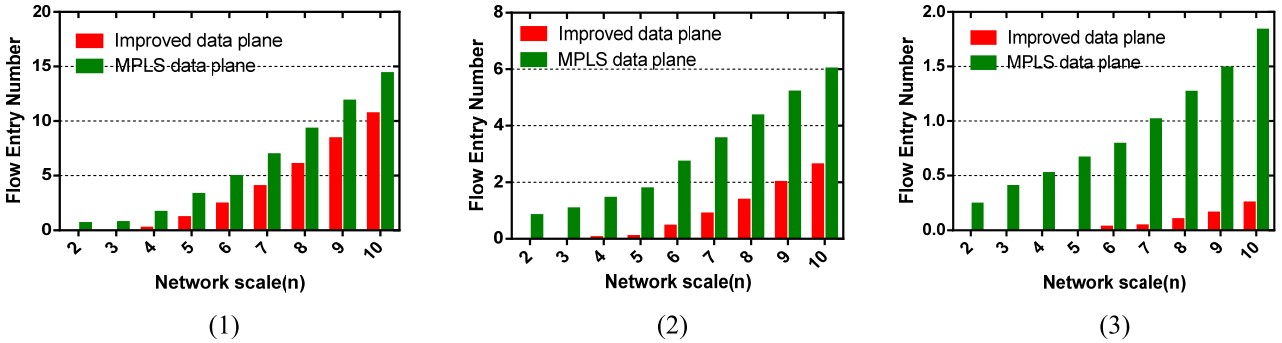


FIGURE 13. Number of flow entries of path encoding using different data plane technologies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

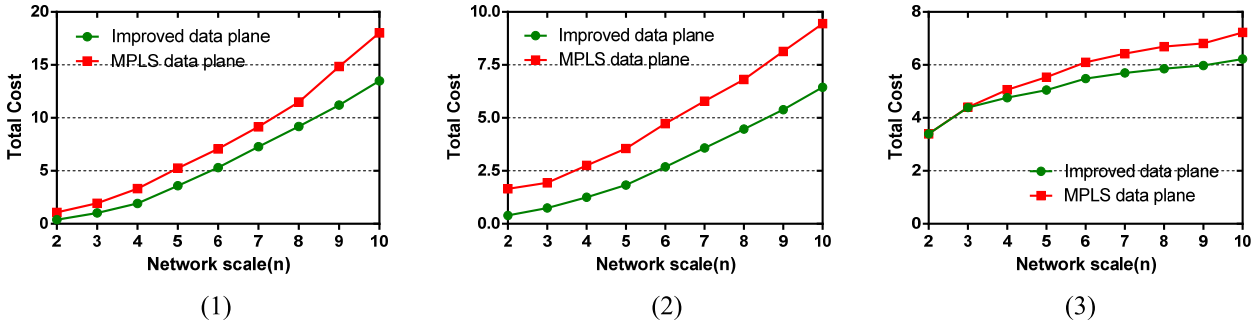


FIGURE 14. Total costs of path encoding using different data plane technologies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

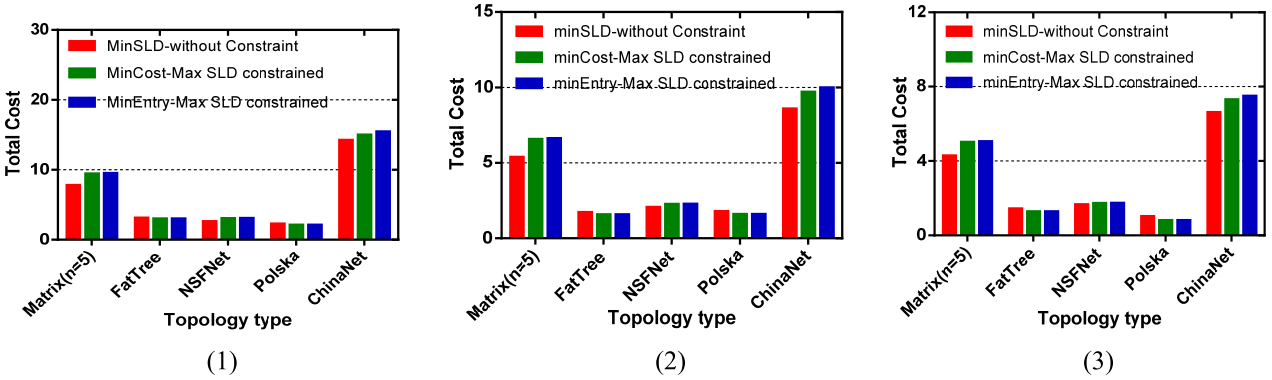


FIGURE 15. Total cost of path encoding under different test topologies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

According to the simulation results, through using the improved data plane, the total cost of path encoding is always less than that using the ordinary MPLS-SR data plane. Moreover, the advantage of the improved data plane increases with the network size.

E. TOTAL COSTS OF DIFFERENT PATH ENCODING STRATEGIES

First, we evaluate the performances of path encoding strategies under different test topologies. According to Fig. 15 and Fig. 16, although the minSLD algorithm can achieve a smaller total cost than the other two algorithms in the Matrix, NSFNet and ChinaNet topologies, its SLD exceeds the constraint value. Therefore, the minCost algorithm can always achieve the lowest cost under the constraint of the maximum SLD, regardless of the topology.

Then, we evaluate the performances of the path encoding strategies under different network scales. According to Fig.17(1) and Fig.18(1), for the MIRA routing algorithm, when the network scale is 2∼7, the minSLD algorithm achieves a smaller cost than the minCost algorithm and the minEntry algorithm because it has no maximum

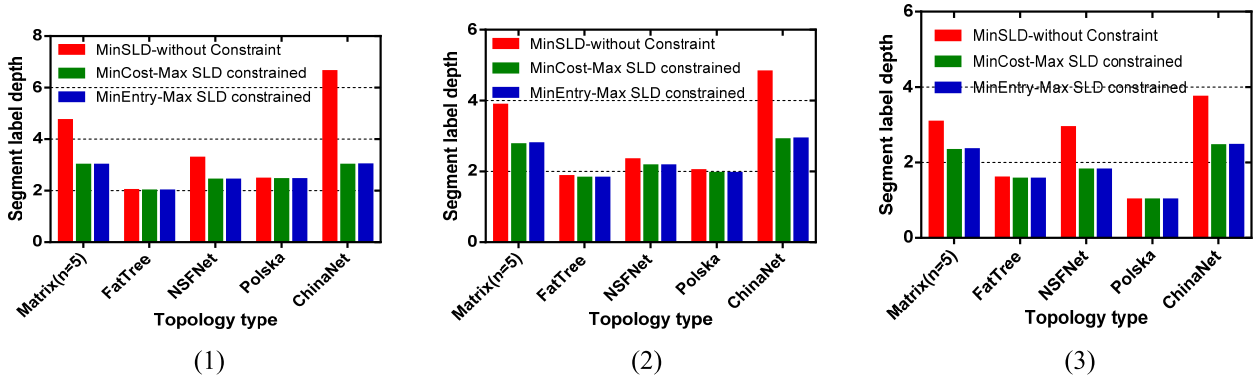


FIGURE 16. Segment list depth under different test topologies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

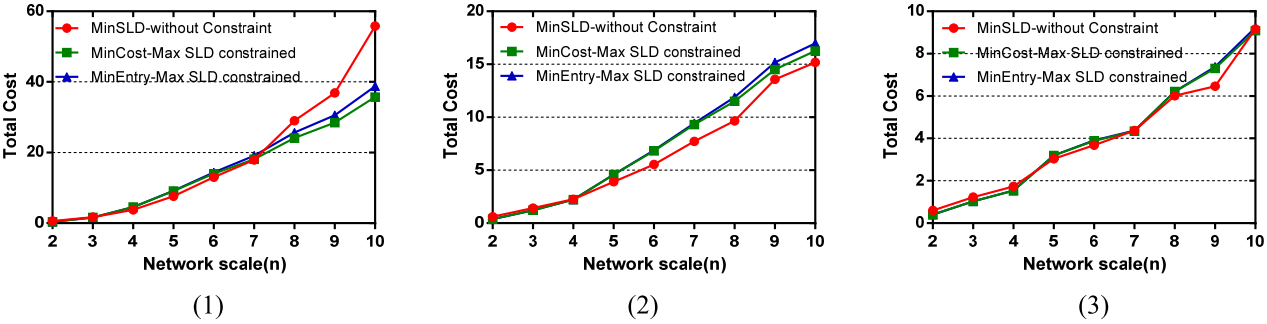


FIGURE 17. Total cost of path encoding using different path encoding strategies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

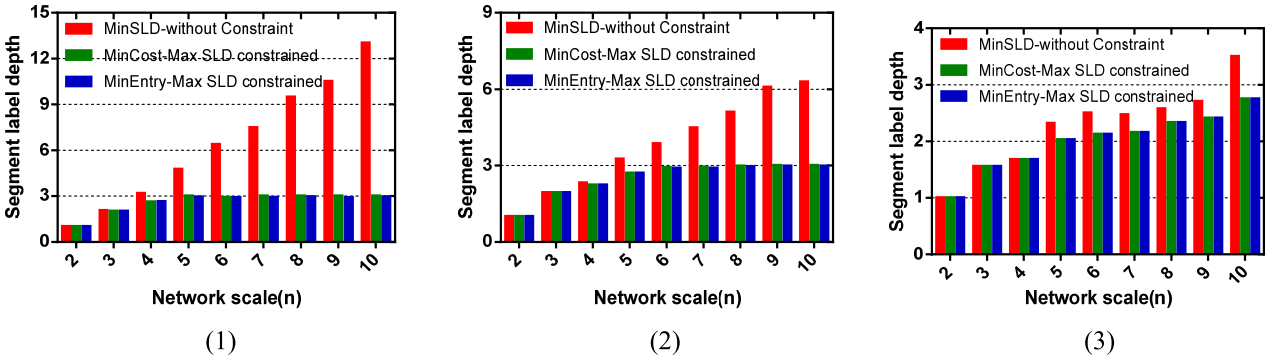


FIGURE 18. Segment list depth using different path encoding strategies. (a) MIRA routing algorithm. (b) WPF routing algorithm. (c) CSPF routing algorithm.

SLD constraint. By increasing the length of the label stack in the packet header, minSLD can effectively reduce the number of new flow entries to be installed, thereby reducing the total cost. With such network scales, the costs of the minCost and minEntry algorithms are almost the same because the flow entry overhead is the main factor that determines the total cost. However, when the maximum depth of the label stack exceeds a specific value (network scale of 8∼10), the traffic overhead becomes the major part of the total cost. As a result, the cost of the minSLD algorithm increases rapidly, while the minCost algorithm has the lowest total cost.

For the WPF routing algorithm, the minSLD algorithm can always achieve the lowest cost since it never requires additional flow entries for fulfilling the maximum SLD constraint. However, the maximum SLD value using minSLD exceeds the limit of 3 when the network scale is more than 5. Meanwhile, comparing with the minEntry algorithm, minCost can always achieve a lower total cost.

For the CSPF routing algorithm, the situation is similar to that of WPF, although the minCost algorithm can always achieve a total cost that is no larger than those of the other two algorithms. However, it has the largest maximum SLD value because it never establishes new flow entries.

From Fig. 15 to Fig. 18, we conclude that the total cost of the minSLD algorithm is not less than the total costs of the other two algorithms when its maximum SLD value satisfies the constraint (less than 3). Meanwhile, when the total cost of the minSLD algorithm is lower, its maximum SLD value also exceeds the constraint. Moreover, comparing with the minEntry algorithm, when the flow entry overhead is the major part of the total cost, minCost can achieve the same cost as the minEntry algorithm. In other cases, minCost can always achieve a smaller total cost.

F. EFFECTIVENESS OF THE ROUTING PLANNING ALGORITHM

Furthermore, we evaluate the performance of the path encoding algorithm when combined with the routing algorithm. The algorithm that we choose is the Constrained Shortest-Path-First algorithm. The scenario in which the path encoding algorithm and the CSPF routing algorithm are executed independently is used for comparison. In the simulation, we use CSPF to represent the result that is obtained when the CSPF algorithm and path encoding are conducted independently and use CSPF-PE to represent the algorithm that was proposed as Algorithm 4. First, we evaluate the two methods under test network topologies as shown in Fig. 19.

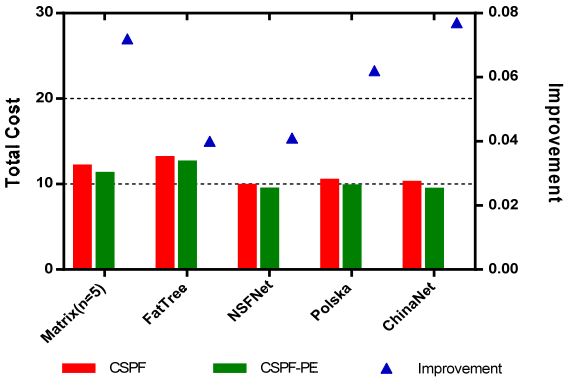


FIGURE 19. Total costs of routing algorithms under different test topologies.

As Fig. 19 shows, regardless of the topology type, the total cost of segment routing can be further reduced by combining the path encoding algorithm with the routing algorithm. In the ChinaNet topology, the effect is the best and the total cost is further reduced by 8%. Moreover, we evaluate how the total cost changes with the network scale in the Matrix network topology, as shown in Fig. 20.

According to the result, for any network scale, the combination of the path encoding algorithm and the routing algorithm can always further reduce the total cost of segment routing and the improvement increases with the increase of the network size.

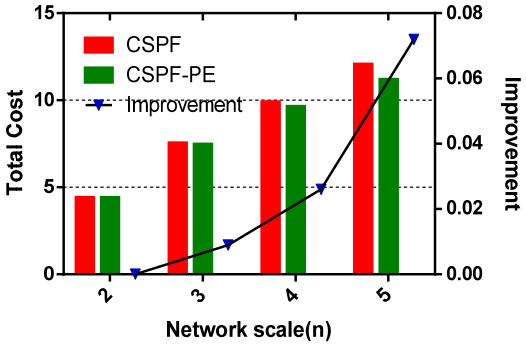


FIGURE 20. Total costs of routing algorithms under different network scales.

G. SUMMARY

Based on the simulation results, we draw the following conclusions:

1) EFFECTIVENESS

Through improving the data plane of segment routing with Openflow technology, the label consumption and the number of flow entries that are used for segment routing under the maximum SLD constraint can be effectively reduced, thereby effectively reducing the total cost of segment routing. On the basis of the improved data plane, by optimizing the path encoding algorithm, not only can the maximum SLD constraint be fulfilled, but also the total cost can be further reduced. Finally, by combining the path encoding algorithm with the routing algorithm, we can further minimize the total cost of segment routing and achieve the best possible performance with the improved data plane and path encoding algorithms.

2) INFLUENCE FACTORS

The performances of the proposed mechanisms and algorithms are impacted by the SLD constraint value, network scale and routing algorithm. The more restrictive the SLD constraint is (the smaller the SLD constraint is), the better the performance that the proposed method can achieve. However, at the same time, the complexity of the network devices will also increase. When the network size is larger, the method that is proposed in this paper is more effective. Therefore, it is suitable for large-scale networks. When the routing algorithm is similar to the LSP establishing algorithm, the method that is proposed in this paper can achieve less improvement.

3) AREAS FOR IMPROVEMENT

Comparing with the traditional path encoding algorithms, the complexity of the proposed algorithm is still high. When it is applied to a complex network or with many users, it may require longer computation time and result in a longer response time of the network service. Therefore, further improvement of the path encoding algorithms is needed. Moreover, comparing with the path encoding algorithms

without maximum SLD constraint, the proposed algorithm need to install new flow entries, which will introduce extra overhead to the network management. Therefore, in the practice, the administrator should decide whether to deploy the proposed algorithm according to the application scenarios and user demands. In the next step of our research, we will address these issues.

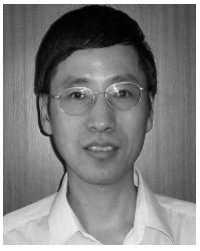
VI. CONCLUSIONS

This paper studies issues of segment routing under the constraint of maximum SLD. To address these issues, we use Openflow technology to improve the MPLS-SR and effectively reduce the label consumption and the number of flow entries of segment routing under the maximum SLD constraint. In addition, to obtain the optimal path encoding solution under the constrained scheme, we present the MaxSLD-PE problem. Formalizations and algorithms are proposed for solving the problem. Through simulations, it is demonstrated that the proposed mechanisms and algorithms outperform the ordinary segment routing solutions with the maximum SLD constraint. In the next step of the research, we will work to reduce the complexity of the path encoding algorithm.

REFERENCES

[1] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *Proc. IEEE Global Commun. Conf.*, Dec. 2015, pp. 1–6.
[2] C. Filsfils *et al.*, "Segment routing with MPLS data plane," IETF draft-ietf-spring-segment-routing-mpls-00, 2014.
[3] S. Previdi *et al.*, "IPv6 segment routing header (SRH)," Internet draft, draft-previdi-6man-segment-routing-header-03, work in progress, 2014.
[4] D. Lebrun and O. Bonaventure, "Implementing IPv6 segment routing in the linux kernel," in *Proc. Appl. Netw. Res. Workshop ACM*, 2017, pp. 35–41.
[5] D. Lebrun, "A linux kernel implementation of segment routing with IPv6," in *Proc. IEEE Comput. Commun. Workshops*, Apr. 2016, pp. 1019–1020.
[6] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based NFV infrastructure," in *Proc. IEEE Netw. Softwarization*, Jul. 2017, pp. 1–7.
[7] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier, "Label encoding algorithm for MPLS segment routing," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, Nov. 2016, pp. 113–117.
[8] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *Proc. IEEE Global Commun. Conf.*, Dec. 2015, pp. 1–6.
[9] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *Proc. Int. Conf. Opt. Netw. Design Modeling*, 2015, pp. 34–38.
[10] A. Cianfrani, M. Listanti, and M. Polverini, "Translating traffic engineering outcome into segment routing paths: The encoding problem," in *Proc. Comput. Commun. Workshops*, 2016, pp. 245–250.
[11] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano, "Traffic engineering with segment routing: SDN-based architectural design and open source implementation," in *Proc. 4th Eur. Workshop Softw. Defined Netw. IEEE Comput. Soc.*, Oct. 2015, pp. 111–112.
[12] A. Sgambelluri, A. Giorgetti, F. Cugini, G. Bruno, F. Lazzeri, and P. Castoldi, "First demonstration of SDN-based segment routing in multi-layer networks," in *Proc. Opt. Fiber Commun. Conf. Exhibit.*, 2015, pp. 1–3.
[13] O. Dugeon, R. Guedrez, S. Lahoud, and G. Texier, "Demonstration of segment routing with SDN based label stack optimization," in *Proc. IEEE Innov. Clouds, Internet Netw.*, Mar. 2017, pp. 143–145.
[14] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. IEEE World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.
[15] R. Carpa, O. Glück, and L. Lefevre, "Segment routing based traffic engineering for energy efficient backbone networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst.*, Dec. 2015, pp. 1–6.
[16] R. Carpa, O. Gluck, L. Lefevre, and J.-C. Mignot, "Improving the energy efficiency of software-defined backbone networks," *Photon. Netw. Commun.*, vol. 30, no. 3, pp. 337–347, 2015.
[17] G. Trimponias, Y. Xiao, H. Xu, X. Wu, and Y. Geng. (Mar. 2017). "On traffic engineering with segment routing in SDN based WANs." [Online]. Available: https://arxiv.org/abs/1703.05907
[18] M. C. Lee and J. P. Sheu, *An Efficient Routing Algorithm Based on Segment Routing in Software-Defined Networking*. New York, NY, USA: Elsevier, 2016.
[19] E. Moreno, A. Beghelli, and F. Cugini, "Traffic engineering in segment routing networks," *Comput. Netw.*, vol. 114, pp. 23–31, Feb. 2017.
[20] S. Bidkar, A. Gumaste, and A. Somani, "A scalable framework for segment routing in service provider networks: The omnipresent Ethernet approach," in *Proc. IEEE Int. Conf. High Perform. Switching Routing*, Jul. 2014, pp. 76–83.
[21] S. Bidkar, A. Gumaste, and A. Somani, "A scalable framework for segment routing in service provider networks: The Omnipresent Ethernet approach," in *Proc. IEEE Int. Conf. High Perform. Switching Routing*, Jul. 2014, pp. 76–83.
[22] S. Salsano, L. Veltri, L. Davoli, P. L. Ventre, and G. Siracusano, "PMSR—Poor man's segment routing, a minimalistic approach to segment routing and a traffic engineering use case," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2016, pp. 598–604.
[23] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure, "SCMon: Leveraging segment routing to improve network monitoring," in *Proc. IEEE Int. Conf. Comput. Commun. INFOCOM*, Apr. 2016, pp. 1–9.
[24] Open Networking Fundation, "Software-defined networking: The new norm for networks," ONF White Paper 2, 2012, pp. 2–6.
[25] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. A World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.
[26] B. Pfaff *et al.*, "The design and implementation of open vSwitch," in *Proc. USENIX SAGE*, vol. 40. 2015, pp. 12–16.
[27] M. Kodialam and T.V. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," in *Proc. IEEE INFOCOM*, vol. 2. Mar. 2000, pp. 884–893.
[28] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, 2nd ed. San Mateo, CA USA: Morgan Kaufmann, 2018, pp. 30–63.
[29] K. Manayya, "Constrained shortest path first," Internet Draft, Feb. 2009. [Online]. Available: http://www.ietf.org/id/draft-manayya-cspf-00.txt
[30] X. Jiang and S. Li. (2017). "Beetle antennae search without parameter tuning (BAS-WPT) for multi-objective optimization." [Online]. Available: https://arxiv.org/abs/1711.02395
[31] L. Jin, S. Li, B. Hu, M. Liu, and J. Yu, "Noise-suppressing neural algorithm for solving time-varying system of linear equations: A control-based approach," *IEEE Trans. Ind. Informat.*, Jan. 2018.

LIAORUO HUANG received the B.E. degree in information security from Xidian University, Xian, China, in 2011, and the M.S. degree in information security science and technology from PLA Army Engineering University, Nanjing, China, in 2014, where he is currently pursuing the Ph.D. degree with the College of Communications Engineering. His research interests include software-defined network, next generation networks, and wide area network.



QINGGUO SHEN received the Ph.D. degree from PLA Army Engineering University, Nanjing, China, in 1994. He is currently a Full Professor with the College of Communications Engineering, PLA Army Engineering University. His research interests include next generation networks, M2M, and wireless networks.

WENJUAN SHAO received the B.E. degree in computer communication and the M.S. degree in computer networks from the Beijing University of Posts and Telecommunications, Beijing, China, in 2001 and 2004, respectively. She is currently pursuing the Ph.D. degree with the College of Communications Engineering, PLA Army Engineering University. In 2007, she became a Senior Lecturer with the Zijin College, Nanjing University of Science and Technology. She was with China Mobile Communications Corporation from 2004 to 2012 as an Engineer. Her research interests include D2D, social aware network, and software-defined network.

CUI XIAOYU received the B.E. degree in mathematics from the Hebei University of Technology, Tianjin, China, in 2016. She is currently pursuing the M.S. degree with the College of Communications Engineering, PLA Army Engineering University. Her research interests include network optimization and network traffic analysis.