Single Tag Scheme for Segment Routing in Software-Defined Network

Wang Shuai, Nattapong Kitsuwan, and Eiji Oki

Department of Communication Engineering and Informatics,
The University of Electro-Communications, Tokyo, Japan.

***Abstract*—This paper proposes a scheme to reduce a size of a packet header for a segment routing (SR) architecture in a software-defined network. In the segment routing architecture, a segment identification (SID) list inserted in packet header is used to indicate a path between source and destination. An SID corresponds to a segment. In some networks, the path may consist of a long SID-list. As a result, a large packet overhead may be occurred and SR may not guarantee the multiprotocol label switching (MPLS) backward compatibility due to that currently deployed MPLS equipments typically support a limited number of stacked labels. In the proposed scheme, the SID-list is replaced by a single tag to indicate the route. A destination IP address of the packet is modified at necessary network elements along the route. To achieve the objective of the proposed scheme, an algorithm to calculate flow entries is introduced. An analytical result shows that the size of the packet header becomes 47% smaller, compared to a scheme that uses the SID-list, when the length of SID-list is 16.**

I. Introduction

Segment routing (SR) [1] is a routing paradigm to provide traffic engineering (TE) by simplifying control plane operations. Instead of using a signaling protocol, an extended interior gateway protocol (IGP), such as open shortest path first (OSPF), is used to advertise segment identifiers (SIDs) [2]. In a multiprotocol label switching (MPLS) network, an SID is presented as a label. A given path is denoted by a stack of SIDs, called an SID-list. The SID-list corresponds to a stack of labels in MPLS architecture [3]. By applying the SID-list only at an ingress node, packets can travel from a source to its destination through a specific path. The configuration of routing at intermediate nodes is not required.

In particular, only the top of the SID-list is effective when packets are executed at an intermediate node. There are basically two types of SID. First, a node-SID identifies a specific node in the network. A packet that has a node-SID on the top of SID-list must be delivered to the corresponding node determined by an equal-cost multi-path (ECMP) routing, which is load balance manner when there exists multi shortest paths to the same destination. Second, an adjacency-SID specifies a link between two adjacent nodes. A packet that has the adjacency-SID on the top of SID-list is delivered to a neighbor node.

Basically, only one SID is needed when the packet travels along a path determined by ECMP. Several SIDs are required with two situations in [4]. First, a natural path is calculated from a basic routing protocol such as OSPF. Second, a TE path is calculated from a constrained shortest path first (CSPF). Constrain of the TE path is either to meet the customer requirements or avoid a congestion. The SID-list becomes longer with more constrains. In general, an SID label needs 4-octets [5] in a packet header. The size of the packet header becomes larger with increasing of the number of SID labels. A packet size, including a header and data, is limited. Fewer data may be carried due to the large size of the header. Therefore, the same amount of the raw data needs more number of packets when the packet size exceeds the limitation. Algorithms in [6] were developed to calculate SID-list with minimum length for pre-determined path. However, under some specific networking scenarios, long SID-list still exists for some paths.

Software-defined networking (SDN) [7] is a centralized control architecture for computer network. In general, it consists of a network controller and network forwarding elements, e.g. switches. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable. An OpenFlow protocol [8] is the most common communication protocol between the SDN controller and the network elements. It provides a series of functions for the controller to modify the information in the packet header using a software.

In this paper, a single tag scheme is proposed to reduce the size of the packet header in SR architecture. By adopting the benefits of the SDN network, a destination IP address of a packet is modified at intermediate switches. The proposed scheme insert an SR tag into the packet header at an ingress node. The destination IP address is changed based on the given SID-list at necessary switches. The packets are forwarded along the specific path by a traditional IP routing, instead of label switching procedure.

The paper is organized as following. In section II, the operation mechanism of SR will be described in detail. In section III, the proposed a single tag scheme and an algorithm for the SDN controller to make the decision of flow entries will be described in detail. In section IV, the proposed scheme will be analyzed. Finally, section V concludes the overall of the paper.

II. Segment routing operation

Segment routing is typically associated with a centralized control plan implementation, [4], [9]. When a traffic flow

has to be established, a request is sent to the controller to compute a path. The controller encodes the computed path into a sequence of SIDs, called an SID-list. In MPLS networks, the SID-list are performed by MPLS rules. At each node, the top SID of SID-list is read, and the packet is forwarded through out an interface associated with the SID in the forwarding table. It is supposed that open shortest path first with traffic engineering extensions (OSPF-TE) is used to advertise the node identifiers, as proposed in [2]. Figures 1 and 2 show a network topology and a forwarding table of node A, respectively.

Assume that a new traffic flow from host $H1$ to host $H2$ ($P_{H1,H2}$) is requested. Without any constrain, the controller selects the path $\{A, E, I, M\}$, since it is the shortest path between $H1$ and $H2$, the SID-list used to encode $P_{H1,H2}$ ($S_{H1,H2}$) has only one SID label, which is $S_{H1,H2} = \{M\}$. $S_{H1,H2}$ is added into the packet header at node A. The SID $\{M\}$ is popped at node I before forwarding the packet to M due to the mechanism named penultimate hop pop (PHP). Then node M will receive a packet without any SID label and send it to $H2$

In the case of a constrain C_1 is applied, the controller may select the path $\{A, B, C, G, H, L, P, O, N, M\}$. A possible SID-list of $S_{H1,H2}$ with a constrain C_1 ($S^{C_1}_{H1,H2}$) may be $S^{C_1}_{H1,H2} = \{C, G, H, P, M\}$. In this case, $S^{C_1}_{H1,H2}$ is added into the packet header at node A, and the packet will be forwarded to node C along the shortest path via node B. And the SID $\{C\}$, which is the top SID on the SID-list, is popped at node B before forwarding the packet to node C. At node C, the SID label $\{G\}$ is popped, and the packet is forwarded to node G. At node G, the SID label $\{H\}$ is popped, and the packet is forwarded to node H.At node H, since the top SID label on the SID-list is P, node H forwards the packet to an output interface that can reach to node P as a shortest path. At node L, SID $\{P\}$ is popped, and the packet is forwarded to node P. Node P and node O will forward the packet to node M along the shortest path and at node N, SID $\{M\}$ will be popped. Here, there is no SID-list in the packet header. As a result, the packet is forwarded to node M and finally reached $H2$ by using the conventional IP routing scheme.

III. Proposed Single Tag Scheme

This section discusses how the proposed scheme works by taking the advantage of the SDN network.

A basic conception of proposed scheme is to modify the destination IP address in the packet header at necessary switches before forwarding packets. An SR tag, which is an identified tag for a pair of source and destination, will be added into the packet header to instruct the specific route so that the network elements can perform the different actions. Consider the same example in section II, the SID-list of $S^{C_1}_{H1,H2}$ is $\{C, G, H, P, M\}$. Based on the encoded SID-list, controller will distribute specific flow entries to associated switches.

Figure 3 shows the contents of flow entries received from controller in each switches along the path. We call them extra flow entries. Based on the flow entries and longest matching principle, when $H1$ sends packets to $H2$, node A receives the

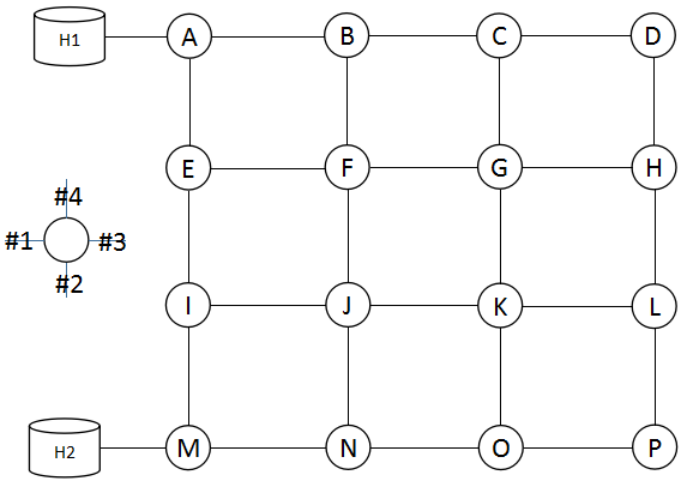

Fig. 1. Example of segment routing.

Node A	
In Label	Action
B	Pop,out3
C	out3
D	out3
E	Pop,out2
F	out2,out3
G	out2,out3
H	out2,ou3
I	out2
J	out2,out3
K	out2,out3
L	out2,out3
M	out,2
N	out2,out3
O	out2,out3
P	out2,out3

Fig. 2. Flow table of node A

packets without a tag in the header. After matching against the flow table with the field of source_IP and dest_IP, the packets from node A are selected and associated actions which consists of adding a tag XY, changing the dest_IP from $H2$ to C, and forwarding to port 3, will be executed. The packets are forwarded to node C. At node B, because no extra flow entries is distributed, so just forward the packets using conventional scheme. At node C, after matching against the flow table with the field of source_IP, dest_IP, and tag, the extra flow entry is selected and associated actions which consists of changing dest_IP from C to G, and forwarding to port 2, will be executed. At node G, after matching against the flow table, the packets are forwarded to port 3 after changing dest_IP from G to H. Node H receives the packets, and forwards them to port 2 after changing the dest_IP from H to P. At node P, the tag XY will be popped and node P forwards the packets to port 1 after changing the dest_IP from P to $H2$.

Node A			
source_IP	dest_IP	Tag	Action
H1	H2		add_tag=XY dest_IP:C output_port:3
Node C			
source_IP	dest_IP	Tag	Action
H1	C	XY	dest_IP:G output_port:2
Node G			
source_IP	dest_IP	Tag	Action
H1	G	XY	dest_IP:H output_port:3
Node H			
source_IP	dest_IP	Tag	Action
H1	H	XY	dest_IP:P output_port:2
Node P			
source_IP	dest_IP	Tag	Action
H1	P	XY	Pop tag dest_IP:H2 output_port:1

Fig. 3. The extra flow entry inserted in each nodes

Finally, the packets are forwarded to $H2$.

From this procedure,we can see it is necessary for the controller to make the decision that what flow entries are inserted into which switches. An algorithm is introduced, using the following notation:

- TOP: the top SID in SID-list
- BOTTOM: the bottom SID in SID-list
- NEXT: the next SID of TOP
- Action_set(SID): a set of action associated with corresponding node
- Add_action(): add additional actions to Action_set(SID)
- Next_IP: an IP address of NEXT
- SID-list-IN: an encoded segment list which ingress node are included (e.g. $S^{C_1}_{H1,H2} = \{A, C, G, H, P, M\}$)
- TAG: tag field in a flow entry
- Src_IP: source field in a flow entry
- Dest_IP: destination field in a flow entry
- INGRESS: ingress node
- Output_port: the output port for forwarding flows
- G: the graph of the network

The pseudo-code of flow entry distribution algorithm are described in Fig. 4.

The algorithm takes as input the graph of topology and an encoded SID-list-IN, and return action_set of each node. It starts with checking if the TOP is the penultimate node or not. If it is the penultimate node and then check if it is the ingress node or not, if it is the ingress node (current top node is both penultimate node and ingress node), then finish. Otherwise (current top node is the penultimate node but not the ingress node), adding corresponding actions, which consists of IP address changing and pop and denoting the output port number, to action_set of this node, and finish. If it is not the penultimate node, then check if this node is ingress node or not, if it is (current top node is not the penultimate node but is the ingress node), adding corresponding actions, which consists of IP address changing and add tag and denoting the output port number, to action_set of this node.Otherwise (current top node is not both the penultimate node and the ingress node), adding actions, which consists of IP address changing and denoting the output port number, to action_set of this node. At last, TOP and NEXT are all pointing to the next SID.

```
function FlowDistribution(SID-list-IN)-->action_set of each nodes
  while(1):
    //check if the next SID is destination or not
    if NEXT is Dest:
        //check if the TOP node is the source node or not by seeing the tag field
        if tag==NULL:
            Break;
        else:
            Action_set(TOP)=Add_action(Dest_IP=Next_IP,untag,Output_port);
            Break;
     //check the TOP is Src or mid node
    if TOP is Src:
        Action_set(TOP)=Add_action(tag=k,output_port,Dest_IP=Next_IP)
    else:
        Action_set(TOP)=Add_action(output_port,Dest_IP=Next_IP)
```

Fig. 4. Flow entry calculation algorithm

IV. Performance Analysis

In the analysis, packet header length of proposed scheme is compared to the conventional SR scheme are calculated. And the ratio of the packet header is illustrated. In addition, the effects of increasing in packet header are also discussed in terms of extra packets and fragment.

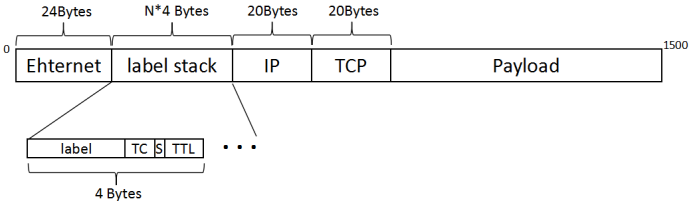

Fig. 5. Ethernet packet format with MPLS label stack

The format of an Ethernet packet is given in Fig. 5. In an Ethernet network, maximum transmission unit (MTU) is set to 1,500 bytes as default. The size of an Ethernet frame is 24 bytes, an IPv4 header (without any options) is 20 bytes, a TCP header (without any options) is 20 bytes. Totally, a packet header has the length of $24 + 20 + 20 = 64$ bytes. The format of an element in label stack consists of 20 bits for the label name, 3 bits the TC field, 1 bit for the S field and 8 bits TTL field, which totally 32 bits (4 bytes). Thus, a packet header

with an SID-list, which N labels are included, has the length of $64 + 4 * N$ bytes. Then the ratio of the header in a packet is illustrated in Fig. 6

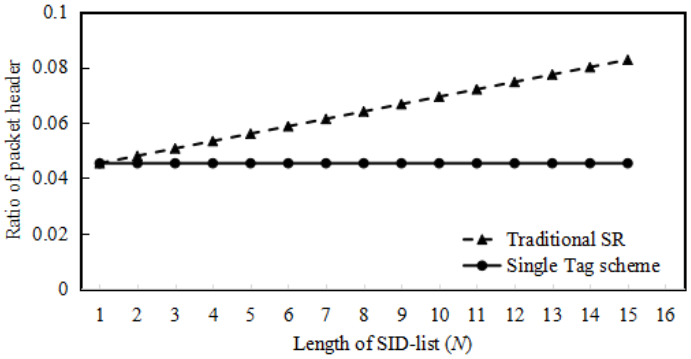

Fig. 6. Ratio of packet header with different SID-list length

We can see that traditional SR takes about nine percents length of MTU and proposed scheme takes about four percents length of MTU when the length of SID-list is 16. Even though the ratio of the header in a packet follows a linear increase with the increase of SID-list, it still should be taken into consideration because of the following reasons.

First, in large scale network, large quantity of data will be forwarded through, even a slight increasing in packet header may cause amount of extra packets forwarding into the network. The total number of packets are following the equation below,

$$Total = \frac{A}{1436 - 4N}, \tag{1}$$

where A is volume of the data file and N is the length of the SID-list. Particularly, in conventional MPLS-TE scheme and proposed single tag scheme, $N = 1$. So the proposed single tag scheme will have the same total number of packets with conventional MPLS-TE scheme when forwarding same amount of data. But in traditional SR, extra packets are introduced due to some data space are used as part of header space for conveying SID-list. The following equations shows the number of extra packets,

$$Total = \frac{A}{1436 - 4N} - \frac{A}{1432}. \tag{2}$$

Figure 7 illustrates how many extra packets will be introduced at the condition which A is 100 Gbytes. As we can see with the 16-length of SID-list, three million extra packets need to be forwarded in the network which may give extra burden to the network.

In addition, when a packet is going into an MPLS domain, by adding a deep SID-list, the total length of the packets may exceed the MTU which can potentially cause fragment procedure which is not wish to happen in network. And for the concern of that currently deployed MPLS equipments typically support a limited number of stacked labels, the proposed scheme can reduce the number of extra packets.

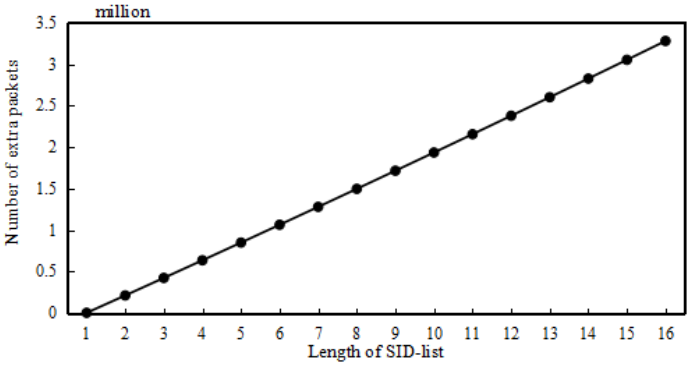

Fig. 7. Extra packets number compare to MPLS-TE

V. Conclusion

In this paper, we proposed a scheme for segment routing architecture to reduce a size of a packet header, and introduced an algorithm for controller to perform flow calculation. By changing the destination IP address along the given path, only one tag is needed to specify the given path, instead of using a SID-list which consists of multi SIDs. Analysis of this scheme indicates that the packet header can be reduce for about 47 percent of the packet header compared to traditional SR when the SID-list is 16 length. And this reduction will increase as the length of SID-list increases which may happen in some large networking scenarios. Our scheme does well specially for the path who has deep SID-list for not introducing extra packets in the network and can avoid fragmentation when packet goes into MPLS domain.

References

[1] C. Filsfils, Ed., et al., "Segment Routing Architecture draft-ietf-spring-segment-routing-07," 2015.

[2] P. Psenak, S. Previdi, et al. "OSPF Extensions for Segment Routing," IETF draft-psenak-ospf-segment-routing-extensions-08, Apr 2016.

[3] C. Filsfils, Ed. et al., "Segment Routing with MPLS data plane," IETF draft-filsfils-spring-segment-routing-mpls-04, 2016.

[4] 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.* Fourth European Workshop on Software Defined Networks (EWSDN), 2015.

[5] E. Rosen, et al., "MPLS Label Stack Encoding," RFC 3032, Jan. 2001.

[6] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, and F. Lazzeri "Path Encoding in Segment Routing," *in Proc.* IEEE Global Communications Conference (GLOBECOM), pp. 1-6, 2015.

[7] "Software-Defined-Network: The new Norm for Networks," ONF White Paper, April 13, 2012.

[8] "OpenFlow Switch Specification," ONF White Paper, April 25, 2013.

[9] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi "Experimental Demonstration of Segment Routing," J. Light. Tech., vol. 34, iss. 1, pp. 205-212, 2016.