Segmented Proactive Flow Rule Injection for Service Chaining Using SDN

Prakash B. Pawar

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

Department of Computer Science and Engineering

June 2016

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Prakash

(Signature)

(Prakash B. Pawar)

CS13M1015

(Roll No.)

Approval Sheet

This Thesis entitled Segmented Proactive Flow Rule Injection for Service Chaining Using SDN by Prakash B. Pawar is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

(T. Bheemarjuna Reddy) Internal Examiner

IITH

[illegible]

(Pravati Swain) External Examiner

NIT Goa

[illegible]

(Dr. Kotaro Kataoka) Adviser

Dept. of Computer Science and Engineering

IITH

[illegible]

(ANTONY FRANKLIN) Chairman

IITH

Acknowledgements

First of all, I would like to express my sincere gratitude to my guide Dr. Kotaro Kataoka for his guidance, valuable feedback and immense knowledge. Without his constant motivation, support and encouragement, I would not have been able to write this thesis.

I would also like to thank all my fellow lab mates for the valuable discussions and fun we had. I am also grateful to Mr.Rahul Patil and Mr. Gaurav Khare who helped in my thesis work.

Finally, I would like to thank friends and family for providing me with unfailing support and continuous encouragement.

Abstract

In today's network, network operator uses different middleboxes to achieve performance and security challenges in the network. Some of the commonly used examples are NAT to solve IPv4 address depletion and firewall to deal the security attacks. These middleboxes comes with new challenges. Service chaining is one of the challenges, where a sequence of middleboxes apply their service to particular traffic. Due to the lack of available protocols to route traffic through middleboxes, operators still rely on error-prone and complex low-level configurations to steer the traffic through the desired sequence of middleboxes. With recent software defined networking (SDN) architecture and OpenFlow protocol, it is possible to steer the traffic through any sequence of middleboxes.

The challenge of service chaining is that a middlebox may alter the content of packet headers; thus, the context of service chaining for particular traffic gets lost. This thesis proposes Segmented Proactive Flow Rule Injection (SPFRI) using SDN. The proposed approach adds inner and outer tag to packets of the flow to maintain the service chain context and next middlebox in the sequence of service chain. Middleboxes need not to be aware of these tags and hence don't need any modification to the operating system or system software of the middlebox. SPFRI maintains the consistency of service chain even though one or more middleboxes may alter the packets at both outbound and inbound directions of a flow. The network configuration of middleboxes must be appropriately set to make them perform in the service chain. The significant benefits of the proposed SPFRI are as follows :

1. No requirement of modifying to the middlebox operating system or software,
2. Steer the traffic through the desired sequence of middlebox in the presence of mangling middlebox,
3. Longer and more chains,
4. scalable and
5. Friendliness to non-SDN switches.

This thesis details the segmentation of the service chain, calculates the required number of flow rules and the algorithm of flow rule calculation to achieve the goals. The thesis discusses required properties, challenges and study of existing service chaining solution. Thesis also discusses the performance of SPFRI and a realistic approach to practical implementation.

Contents

Declaration ii

Approval Sheet iii

Acknowledgements iv

Abstract v

Nomenclature **vii**

1 Introduction **1**

2 Related Work **3**

3 Segmented Proactive Flow Rule Injection **4**

3.1 Double Tagging for Service Chaining 4

3.2 Segmenting a Service Chain 4

3.3 Required Network Configuration for Middleboxes 5

3.4 SPFRI 5

4 System Design and Architecture **7**

4.1 Policy Database 7

4.2 Middlebox Database 8

4.3 Policy Enforcer 8

4.4 SPFRI Engine 8

4.5 Path Finder 8

4.6 Types of Segments 8

4.6.1 Ingress Segment 9

4.6.2 Egress Segment 10

4.6.3 Intermediate Segment 10

4.7 Implementing Inner and Outer Tags 10

4.8 Practical Network Configuration for Middleboxes 11

4.9 Segmented Proactive Flow Rule Injection for a Segment 11

5 Protocol Behaviour **12**

5.1 Segment and Switch Types 12

5.2 Execution of Flow Rule Injection 13

6 Discussion **17**

6.1 Overhead of Double Tagging and Choice of Outer Tag 17

6.2 Mode of Proactive Flow Rule Injection: Partial or Full ? 17

6.2.1 Time Required to Install Flow Rules 17

6.2.2 Dynamic Policy Update 18

6.2.3 Total Number of Flow Rules 18

6.2.4 Limitations 18

7 Evaluation **19**

8 Conclusion and Future Work **21**

References **22**

Chapter 1

Introduction

Various network appliances are widely used for Internet services, enterprise networks and cloud computing environments. Typical examples of network appliances are firewalls, content filters, intrusion detection systems, deep packet inspection, web proxies, load balancers, network address translation (NAT) and transmission control protocol optimizers. Such network appliances are generally referred to as middleboxes or inline services because end users are often unaware of their existence. We need middleboxes in order to achieve performance and security challenges in the network (e.g., consider proxy which stores the content and provides faster for next time, NAT to solve the problem of IP address space depletion and Firewall to deal with various attacks). Service chaining is required when traffic must traverse more than one middlebox in a specific sequence (e.g., web traffic should be processed by a web proxy and then a firewall). Although a significant amount of work has been done in recent years [1] [2] [3] there is still no satisfactory solution to the problem of directing traffic through the desired sequence of middleboxes. To enable service chaining when multiple sequences of middleboxes are involved, the traffic must flow through the right sequence. In addition, middleboxes may alter the context of the packet header. As a result, the context of the service chain is lost. This context is critical to determine the next middlebox in the service chain (e.g., NAT can change the source IP address, source port or both). As these middleboxes are closed and proprietary, problem becomes more difficult.

Service chaining requires carefully planned network topology, a consistent set of rules to route traffic through the desired sequence of middleboxes and safeguards for correct operation in case of failures and overload. Such setups are complex and rigid, often lead to misconfigurations and errors. A lack of protocols and tools to perform accurate configurations makes service chaining complex. SDN offers a promising alternative for service chain implementation. It uses logically centralized management, decouples the data and control plane and provides programmability for forwarding traffic.

We propose SPFRI, a simple but effective and immediately deployable solution to the service chaining problem. SPFRI, which does not require changing the middlebox software or OpenFlow protocols [4], has the following features:

1. Retains the context of the service chain,

2. Consistently supports both outbound and inbound traffic,

3. Maintains the context in a lightweight manner and
4. Supports longer chains.

In SPFRI, the mechanism to divide the end-to-end path of the service chain into **Segments** contributes to the determination of appropriate SDN switches to perform flow rule injection. Minimum restriction to the network configuration of middleboxes is important to preserve the context of the service chain, which is represented by a VLAN ID in 802.1AD [5] as an inner tag for the packet. Given the flexibility of SDN, using VLAN IDs to indicate the service chain and next middlebox, rather than isolating Layer 2 segments.

This thesis describes the network configuration of middleboxes that is appropriate for SPFRI and verify that it enables service chaining without losing context. This thesis also discusses the performance and scalability of SPFRI as well as the advantages and disadvantages of proactive and reactive flow rule injection.

Chapter 2

Related Work

The network service header (NSH) [6] attempts to solve the service chaining problem by adding an additional header to each packet, which can be used by middleboxes for traffic steering. However, this approach requires modifying the middleboxes to make them aware of the additional headers.

StEERING [1] utilizes a pipeline feature introduced in OpenFlow 1.1 for service chaining. It also examines how to select the best locations for placing services in the network to optimize performance. However, it does not clarify how to deal with cases where middleboxes alter the packet header on both outbound and inbound directions or how to ensure flow processing in a sequence of middleboxes.

SIMPLE [3] offers a heuristic correlation approach to maintain the context of service chaining even if a middlebox modifies the contents of a packet header. It requires multiple packets to be sent to the controller before and after processing by a middlebox to identify the flow and its error rate is 19%. This process introduces high computational overhead because multiple packets per flow must be processed by the controller in a stateful manner.

FlowTags [2] employs simple extensions to middleboxes to handle additional tags in packet headers. The tag associates the contextual information of a corresponding service chain with a traffic flow. One drawback is that this requires middlebox modifications for tag generation and consumption at middleboxes. Rewriting packet headers at every middlebox also incurs additional overhead.

Position [7] has different types of drawbacks. Service chain scalability is proportional to the number of users. A new service chain instance must be created for each new user; therefore, the number of flow rules can easily exceed the switch memory capacity. For small duration flow, creation and destruction of the service instance occurs very frequently, which introduces significant overhead on computation and flow rule management on the switch. If the SDN switches are software-based, memory consumption per flow rule will be less when using the proposed SPFRI because the size of the match fields (16 bits of an outer tag using a VLAN ID) is smaller compared to [7] (48 bit MAC address). Such a difference will introduce an impact when the number of flow rules increases.

OpenSCaas [8] uses a source MAC address to encode the service chain context. However, this raises security concerns in the data link layer due to ARP spoofing. Thus, MAC-based access control cannot be implemented.

Chapter 3

Segmented Proactive Flow Rule Injection

The proposed SPFRI offers a solution to service chaining problem. The prerequisites and technical components to enable service chaining and the SPFRI mechanism is described here.

3.1 Double Tagging for Service Chaining

Whenever a packet belonging to a new flow arrives at the ingress switch, a service chain ID is assigned by the SDN controller. All packets in the flow contain a tag to indicate the ID as an additional field. The service chain ID indicates the sequence of middleboxes the packet must go through. SPFRI uses double tagging with an Inner and Outer Tag. The Inner Tag is implemented as the VLAN ID to indicate service chain ID. The advantage of using a VLAN ID as the Inner Tag is the fact that the VLAN ID will not be changed if the packet goes into and comes out of the same VLAN interface of the middlebox. Therefore, even though a middlebox such as NAT or a proxy, modifies the packet header, the VLAN ID will remain the same and the context of the service chain will not be lost. Another advantage is that VLANs are very well supported by the host operating systems of middleboxes and SDN and non-SDN switches. If a middlebox is connected to only SDN switches, our solution will work successfully in a hybrid network where both types of switches coexist. An Outer Tag indicates the next middlebox that the packet must go through in the service chain. The Outer tag is added and removed by the SDN switch that connects to the middlebox. This operation means that middleboxes in the service chain do not have to maintain or understand the context of the service chain. After a middlebox outputs the processed packet using the same VLAN ID, the SDN switch attaches the updated Outer Tag to indicate the next middlebox.

3.2 Segmenting a Service Chain

In a service chain, an altering middlebox may communicate using IP with the following nodes:

1. the original source host of a flow,
2. the adjacent altering middlebox in the service chain and

3. the destination host that the original source host intended to reach.

Once a packet comes out of a middlebox, the content of the MAC header will be consistent until it arrives at another middlebox.

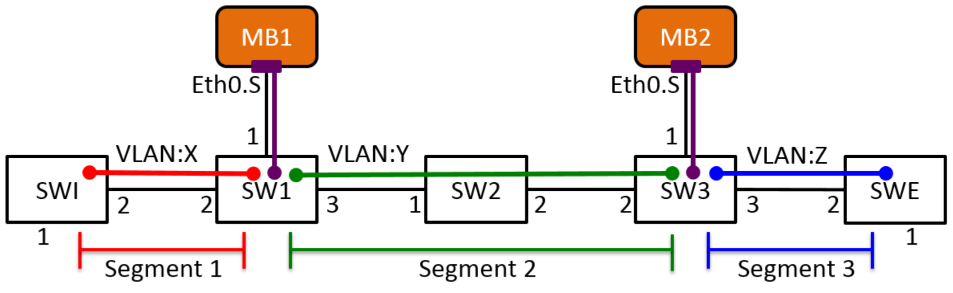

Figure 3.1: Expected Network Diagram for Designing System

Here, the concept of **Segment** is introduced to divide a service chain into sub connections, as shown in Fig. 3.1. An Outer Tag represents the segment. A segment is an intermediate path between switches that are directly connected to middleboxes or the communicating host in a given service chain. For example, Segment 2 (Fig. 3.1) is configured over two connections between SW1 and SW2 and between SW2 and SW3. Once a packet enters the segment from a middlebox, e.g., MB1, the context of the service chain and the five-tuple information (SrcIP, DstIP, SrcPort, DstPort, Proto) of the packet are consistent until it reaches the next middlebox, i.e., MB2. At the end of each segment, the switch, i.e., SW3 for Segment 2, will remove the Outer Tag and update the destination MAC address of the packet so that it can be processed by MB2.

3.3 Required Network Configuration for Middleboxes

The network configuration on the middlebox is a critical part of our solution. Each middlebox configures one or more IP addresses on the same VLAN interface if the middlebox is visible at Layer 3. This configuration ensures that the middlebox is a part of a segment that corresponds to the service chain. Accordingly, no matter how a packet is altered or forwarded to another subnet, this middlebox service will still be provided on the same VLAN.

3.4 SPFRI

The end-to-end path between the communicating source and destination consists of two or more segments in a chain. When the processed packet comes out of the middlebox, the default action on the directly connected SDN switch is to send the incoming packet to the SDN controller. The controller records the flow and corresponding service chain ID and then adds an Outer Tag, i.e., the next middlebox in a service chain. The controller installs flow rules so that SDN switches in a segment can forward the packet based on the Outer Tag. The Controller also proactively installs the flow rules for the reverse direction of the flow on all switches in the same segment. The flow rules forwarding the packets are based on the Outer Tag for the forward direction, and the corresponding Outer Tag for the reverse direction. This proactive flow rule injection maintains the consistency of service chaining in both directions because it is performed together immediately after the packet is altered by the middlebox.

Chapter 4

System Design and Architecture

The proposed system has two major components:

1. the SDN controller for implementing SPFRI and

2. the middleboxes with appropriate network configuration to cope with SPFRI.

This section focuses on the system architecture of the SDN controller. Note that the middleboxes are explained in the Section 4.8. Fig. 4.1 shows the system diagram of SPFRI on the SDN controller.

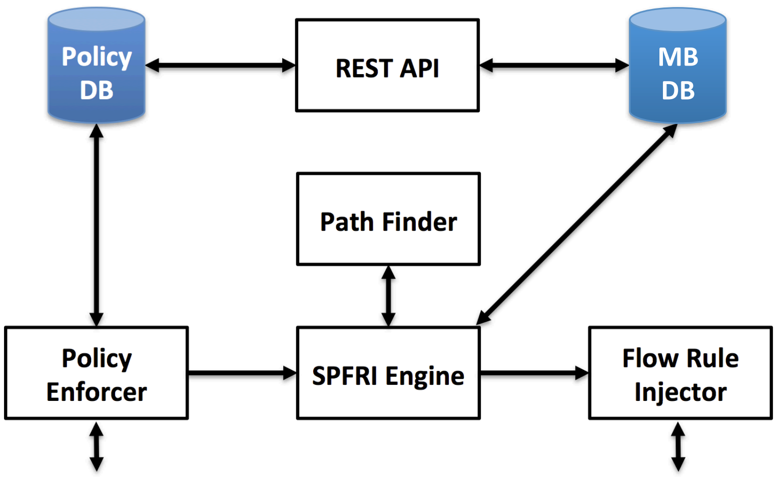

Figure 4.1: System Diagram

4.1 Policy Database

First Network Administrator defines service chaining policy through REST API and stores in the policy database. Policy Database consists of Source IP, Destination IP, Source Port, Destination Port, Protocol and Policy ID or Service Chain ID used interchangeably. Incoming Packet is matched against the policy database to find the Policy ID. Policy ID specifies what are the sequence of middleboxes traffic has to go through. We are considering middleboxes which are both virtualised and physical entities. To implement the policy engine we can consider interoperability with existing policy specification languages such as Frenetic [9], PonderFlow [10], Ponder2 [11]. Policy Enforcer focuses on which policy should be applied to given flow and enforced it through SPFRI.

4.2 Middlebox Database

As we are treating Middleboxes as both physical and virtual entities network administer has to provide this information through REST API. It consists of Middlebox ID, Mac Address, Switch DPID, Switch Port and Ordered List OFAction. In a pure virtualised environment, the NFV controller must provide middlebox specific information which is stored in the middlebox database, to the SDN controller. One approach proposed in [8] can be used where SDN topology manager communicates with NFV topology Manager.

4.3 Policy Enforcer

When the first packet of a new flow reaches an ingress switch from the source host, a packet-in message is sent to the SDN controller. The policy enforcer is responsible for determining the actions that are taken for the flow. The Policy Enforcer communicates with policy database to classify the flow quickly and the service chain to be applied. It also receives packet-in from the switches connected to a middlebox with VLAN header containing service chain ID. The policy enforcer provides the service chain ID and the next middlebox to travel to in the sequence to the SPFRI engine. The further process is to attach the appropriate tag to the packet and send flow mod out .

4.4 SPFRI Engine

The SPFRI engine is responsible for installing flow rules for a given segment. It communicates with the Path Finder to obtain the segment. It also interacts with the middlebox database to install middlebox specific actions on switches connected to middlebox. Then, the appropriate tag is attached to the flow and send flow mod out.

4.5 Path Finder

The path finder determines the sequence of switches to be followed to reach the next middlebox in the chain. It uses the IRoutingService of Floodlight [12] and provides the shortest path to the next middlebox. Note that QoS can be the purpose of service chaining and a part of the service chain. We can flexibly select different paths using the path finder module to implement QoS.

4.6 Types of Segments

Based on the sequence of middleboxes to go through, SPFRI sets up the segment within which the flow rules for packet forwarding are proactively injected for both forward and reverse directions.

To perform proactive flow rule injection, SPFRI in the SDN controller must know which switch plays what role in the given segment. Once SPFRI knows what role each switch plays in the given segment, the service chain can be applied to a flow. SPFRI looks up the mapping between the corresponding middlebox and switch in the middlebox database. In the rest of this section, the switch IDs, which appear in the following descriptions, correspond to those in Figs. 4.2 and 4.3 according to the direction.

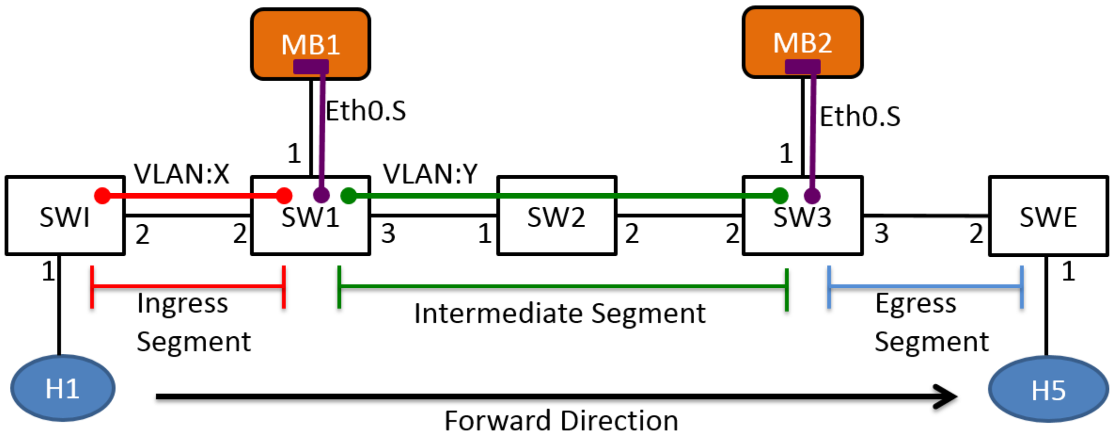

Figure 4.2: Forward Direction Segment VLAN

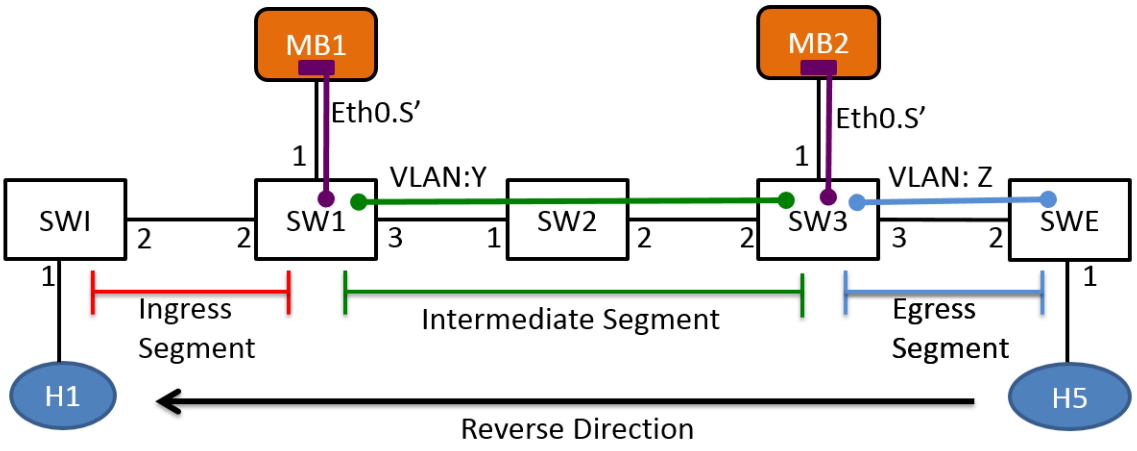

Figure 4.3: Reverse Direction Segment VLAN

4.6.1 Ingress Segment

When the first packet is transmitted by the user, the ingress segment is formed with the ingress switch and all other switches on the path to the first middlebox. SPFRI on the SDN controller triggers proactive flow rule injection, which contains slightly different actions according to the role of the switch. The ingress switch functions as a starting point of a segment.

The **Ingress Switch (SWI)** adds an Inner Tag that corresponds to the service chain ID and an Outer Tag that specifies the next middlebox in the service chain and the output port is determined based on the Outer Tag. For the reverse direction, SWI simply forwards the packet to the destination MAC address in the Ethernet header. The corresponding flow rules are injected for both directions simultaneously.

At **the end point of the Segment**, the switch (SW1) removes the Outer Tag and executes the middlebox specific action on the packet to be processed by the middlebox, such as setting the destination MAC address in the Ethernet header and sending it through the corresponding port. All this required information is stored in the middlebox database. For the reverse direction, the switch removes the Inner Tag because all services from the service chain have been traversed. The switch also modifies the destination MAC address to that of the end host and outputs the packet from the port such that the destination MAC address is reachable.

If **an intermediate switch** exists in the segment, it forwards a packet based on the Outer Tag for the forward direction. For the reverse direction, the switch forwards a packet to the end host just like a learning switch.

4.6.2 Egress Segment

A switch that is directly connected to the last middlebox of the service chain (SW3), is the starting point of the egress segment in the forward direction and the end point in the reverse direction. Accordingly, the egress switch of the flow (SWE) will be the end point of the segment in the forward direction and the starting point in the reverse direction. Here, the flow rules that are proactively injected to these switches are exactly symmetric to the case of the ingress segment.

4.6.3 Intermediate Segment

The intermediate segment starts at a switch directly connected to the corresponding middlebox, e.g., SW1 and MB1. On this switch, SPFRI adds an Outer Tag to the incoming packet to specify the next middlebox in the service chain and outputs it from the port such that the next middlebox is reachable. For the reverse direction, the switch removes the Outer Tag and executes the middlebox specific action on the packet to be processed by the middlebox. If an intermediate switch (SW2) exists in the segment, it forwards a packet based on the Outer Tag for both forward and reverse directions. At the end point of the segment (SW3), the switch removes the Outer Tag and modifies the destination MAC address to that of the VLAN interface of the middlebox and outputs the packet. For the reverse direction, the switch adds an Outer Tag that specifies the next middlebox in the service chain and outputs the packet from the port such that the next middlebox is reachable.

4.7 Implementing Inner and Outer Tags

The VLAN ID field in the VLAN header [13] is 12 bits. We use the VLAN ID field for the Inner Tag with 1 bit for direction and 11 bits are used to identify 2048 different service chains. Similarly, the VLAN header or MPLS header [14] can be used for the Outer Tag. As shown in Table 4.1, the VLAN header as the Outer Tag will get 12 (VLAN ID) +3 (PCP) bits for the middlebox ID, i.e., 32768 different middleboxes in a network. With the MPLS header as the Outer Tag will get 20 (MPLS Label) +3 (QoS) bits for middle box ID, i.e., 8388608 different middleboxes in a network. Note that there is virtually no restriction on the length of the service chain.

Header	Fields	Bits	Middleboxes
VLAN	VLAN ID	12	4096
	VLAN ID + PCP	12 + 3	32768
MPLS	MPLS LABEL	20	1048576
	MPLS LABEL + QoS	20 + 3	8388608

Table 4.1: Outer Tag Implementation

4.8 Practical Network Configuration for Middleboxes

Figure 4.4 shows the configuration of a VLAN interface on a middlebox that is serving as a NAT router. The middlebox configures both internal and external IP addresses on the same VLAN interface using an alias. Even if the NAT router alters the content of packet headers, incoming and outgoing packets have the same VLAN ID in the VLAN header to indicate that they belong to the same segment.

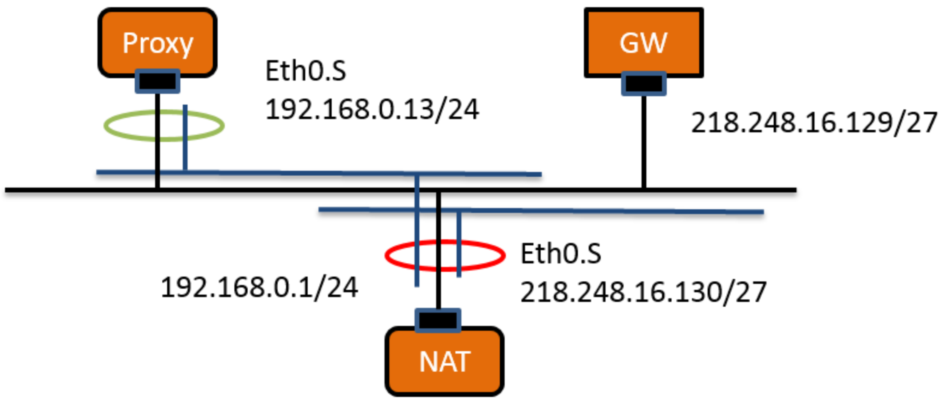

Figure 4.4: Practical Network Configuration for NAT Middlebox

4.9 Segmented Proactive Flow Rule Injection for a Segment

Whenever the first packet of the new service chain ID arrives at the switch from the middlebox, it is sent to the controller with the VLAN (i.e., Inner Tag) header as a packet-in message. Upon receiving the packet with the VLAN tag, SPFRI decodes the tag to find the current and next middleboxes in the service chain for the flow of the same packet. It finds the path between these two middleboxes using the middlebox database. It then installs a flow rule to add an Outer Tag as that of the next middlebox ID on the switch from which the packet-in message is generated. It also installs an Outer Tag based flow rules on all the switches between these two middleboxes to steer the traffic in the forward direction. For the reverse direction, it uses the current middlebox ID as the Outer Tag to steer the traffic in the reverse direction. Note that the rules for the reverse direction flow are installed at the same time.

Chapter 5

Protocol Behaviour

This chapter describes in detail execution of SPFRI.

5.1 Segment and Switch Types

Based on the sequence of middleboxes to go through, SPFRI setups the segment within which the flow rules for the packet forwarding are proactively injected for both forward and reverse directions. Table 5.1 and Table 5.2 show the segments formed in forward and reverse direction. There are three types of the segment.

1. **Ingress Segment:** Formed between Ingress switch and all other switches on the path to the first middlebox in the service chain.
2. **Egress Segment:** Formed between switch connected to last middlebox and all other switches on the path to the end host communicating.
3. **Intermediate Segment:** Formed between switches connected to adjacent middlebox and all other switches on the path between them.

In the segment, there are three types of switches and each of them has a different role and Flow Rule installed.

1. **First Switch:** It adds the Outer tag which next middlebox in Service Chain. In the case of ingress segment, it also adds inner tag corresponding to the service chain id. In the case of egress segment, it removes inner tag as all the middleboxes in the service chain has been traversed.
2. **Intermediate Switch:** It does forwarding based on Outer tag. In the case of partial proactive flow rule injection, it sends the packet to the controller in order to find the next middlebox in the service chain sequence.
3. **End Switch:** It removes an outer tag and executes middlebox specific action on the packet in order to packet being processed by the middlebox like setting mac address in destination address field and sending through the correct output port.

No	Segment Type	Path of Switches	First Switch	Intermediate Switches	Last Switch
1	Ingress	SW1-SW2-SW3	SW1	SW2	SW3
2	Intermediate	SW3-SW2	SW3	-	SW2
		SW2-SW3-SW4-SW5	SW2	SW3, SW4	SW5
3	Egress	SW5-SW6	SW5	-	SW6

Table 5.1: Segments in Forward Direction

No	Segment Type	Path of Switches	First Switch	Intermediate Switches	Last Switch
1	Ingress	SW6-SW5	SW6	-	SW5
2	Intermediate	SW5-SW4-SW3-SW2	SW5	SW4, SW3	SW2
		SW2-SW3	SW2	-	SW3
3	Egress	SW3-SW2-SW1	SW3	SW2	SW1

Table 5.2: Segments in Reverse Direction

5.2 Execution of Flow Rule Injection

To perform the proactive flow rule injection, SPFRI in the SDN controller needs to know which switch plays what role in the given segment. Because once the SPFRI knows what role each switch plays in the given segment, the service chain can be applied to a flow. SPFRI looks up the mapping between the corresponding middlebox and switch in Middlebox Database. In the rest of this section, the switch IDs, that appear in the following descriptions, correspond to those in Figures 5.1 and 5.2 and 4.3 according to the direction.

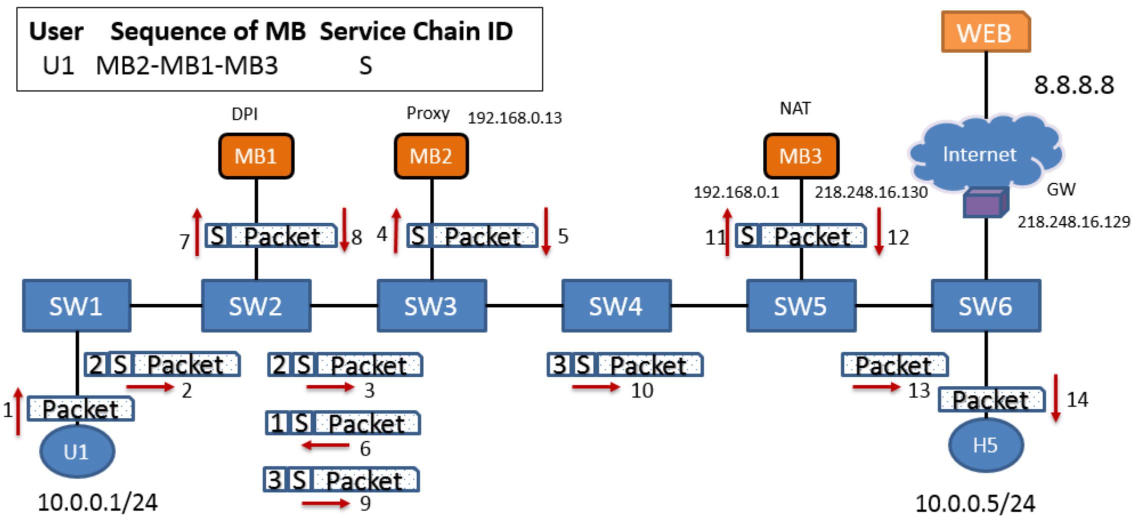

Figure 5.1: Detailed Protocol Behaviour in Forward Direction

Ingress Segment

When the first packet is transmitted by the end host (Fig. 5.1)(1), the ingress segment is formed with the ingress switch and all other switches on the path to the first middlebox. SPFRI on the SDN controller triggers the proactive flow rule injection, which contains slightly different actions according to the role of the switch.

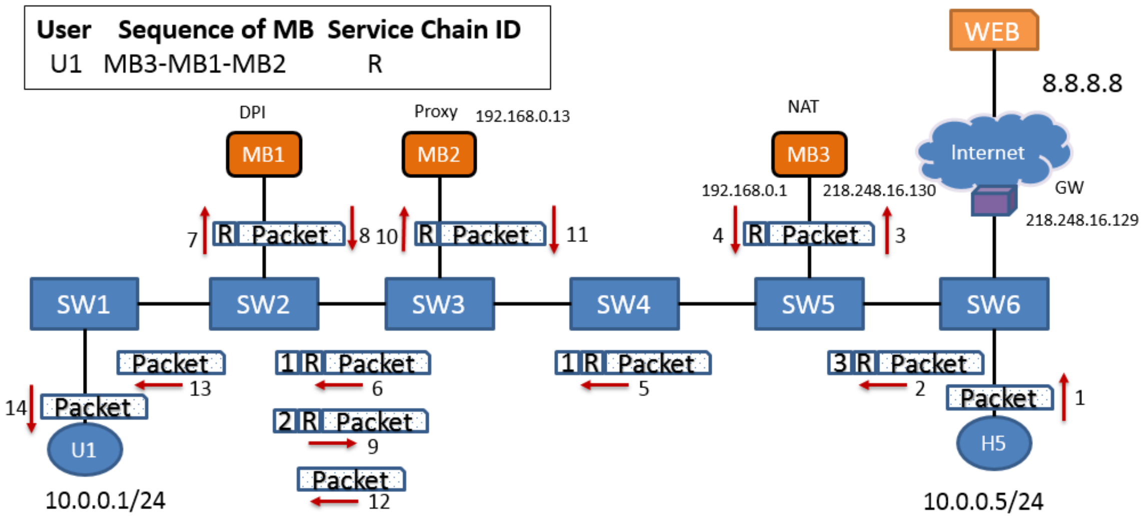

Figure 5.2: Detailed Protocol Behaviour in Reverse Direction

The ingress switch functions as a starting point of a segment and flow rules, with following actions, are installed on the ingress switch.

Forward direction

1) attach Inner Tag corresponding to service chain id and Outer Tag corresponding to 1st middlebox in sequence to the packet, and

2) output the packet from the port to the first middlebox.

As shown with (Fig. 5.1)(1) and (Fig. 5.1)(2).

Reverse direction

1) output the packet to the MAC address of the destination host.

As shown with (Fig. 5.2)(13) and (Fig. 5.2)(14).

The flow rule is determined by the simple learning switch module because of following 1) this ingress switch is directly connected to the destination host and
2) the Inner Tag is removed before reaching this ingress switch.

The flow rule with following actions is installed on the switch at the end point of a segment (SW3).

Forward direction

0) for Outer Tag corresponding to the Middlebox ID [Match Condition],

1) change the destination MAC address to that of the middlebox's (MB1) VLAN interface in the Ethernet header,

2) remove Outer Tag and

3) output the packet to the middlebox.

As shown with (Fig. 5.1)(3) and (Fig. 5.1)(4).

Reverse direction

0) for Inner Tag corresponding to the Service Chain ID with the input port connecting to the last middlebox [Match Condition],

1) remove Inner Tag also from the packet,

2) change the destination MAC address to that host and

3) output the packet from the port through that the MAC address of the destination host is reachable.

As shown with (Fig. 5.2)(12) and (Fig. 5.2)(13).

Note that the flow rule for the reverse direction is based on the expectation that the packet has already gone through all middlebox sequence. Therefore, this particular switch (SW3) removes the tagged header and then delivers it to the destination MAC address according to the learning switch module in SPFRI.

If an intermediate switch exists in the segment, the flow rule with following actions is installed.

Forward direction

0) for Outer Tag corresponding to the middlebox ID [Match Condition],

1) output the packet from the port to the middlebox.

Reverse direction

1) output the packet from the port through that the MAC address of the destination host is reachable.

As shown with (Fig. 5.2)(12) and (Fig. 5.2)(13).

Egress Segment

The switch, which is directly connected to the last middlebox of service chain (SW5), is the starting point of the egress segment in a forward direction as well as the end point in a reverse direction. Accordingly, the egress switch of the flow (SW6) will be the end point of the segment in the forward direction as well as the starting point in the reverse direction. Here, the flow rules that are proactively injected to these switches the exactly symmetric to the case of ingress segment.

Intermediate Segment

The intermediate segment starts at the switch, which is directly connected to the corresponding middlebox (MB1),i.e., (SW2). On this switch, SPFRI gives two different flow rules with a different priority to the switch.

Forward direction: Flow Rule 1 with Low Priority

0) for the first packet of a new flow which comes from port connected to the middlebox [Match Condition],

1) send the packet to the controller.

Note that this action is necessary to let the controller find the next middlebox of the flow corresponding to this service chain.

Forward direction: Flow Rule 2 with High Priority

0) for the packet coming from the middlebox using Inner Tag corresponding to the service chain, which was given by the controller in response to Flow Rule 1 [Match Condition],

1) add Outer Tag corresponding to next middlebox ID and

2) output the packet from the port reachable to the next middlebox.

As shown with (Fig. 5.1)(9) and (Fig. 5.1)(10).

Reverse direction

0) for the packet comes from the middlebox using Inner Tag corresponding to the service chain[Match Condition],

1) change the destination MAC address to that of the middlebox's VLAN interface in the Ethernet header,

2) remove Outer Tag to the one that is configured on the middlebox and

3) output the packet to the middlebox.

As shown with (Fig. 5.2)(5) and (Fig. 5.2)(6).

The flow rule with following actions is installed on the switch at the end point of the intermediate segment (SW2).

Forward direction

0) for Outer Tag corresponding to the middlebox id [Match Condition],

1) change the destination MAC address to that of the middlebox's VLAN interface in the Ethernet header,

2) remove OuterTag that is configured on the middlebox, and

3) output the packet to the middlebox.

As shown with (Fig. 5.1)(6) and (Fig. 5.1)(7).

Reverse direction

0) for the packet comes from the middlebox using Inner Tag corresponding to the Service Chain ID in the reverse direction [Match Condition],

1) add Outer Tag corresponding to next middlebox id and

2) output the packet from the port reachable to the next middlebox. As shown with (Fig. 5.2)(8) and (Fig. 5.2)(9).

The flow rule with following actions is installed on an intermediate switch in the intermediate segment (SW3).

Forward direction

0) for Outer Tag corresponding to the middlebox id [Match Condition],

1) output the packet from the port to the next middlebox.

As shown with (Fig. 5.1)(9) and (Fig. 5.1)(10).

Reverse direction

0) for Outer Tag corresponding to the middlebox id [Match Condition],

1) output the packet from the port to the next middlebox.

As shown with (Fig. 5.2)(5) and (Fig. 5.2)(6).

Chapter 6

Discussion

6.1 Overhead of Double Tagging and Choice of Outer Tag

Each packet in a flow is tagged with an additional Inner and Outer Tag to maintain the context of the service chain. This causes an overhead of 8 bytes for each packet of the flow if the VLAN header is used as both the Inner and Outer Tags. This approach is friendly with existing Layer 2 switches because they also understand VLANs and can be configured in truncated mode to forward packets tagged by SPFRI. Given the benefits achieved by using an 802.1AD header and the active use of VLANs in most enterprise networks, the impact of this additional header can be considered negligible. SPFRI also helps the deployment of service chaining in a hybrid network environment that consists of SDN and non-SDN switches. For the VLAN as Inner Tag and MPLS as an Outer Tag, the overhead is the same, i.e., 8 bytes. Using a packet header compression technique, the overhead of the tag for small packets can be reduced; however, the processing overhead at the controller may increase.

6.2 Mode of Proactive Flow Rule Injection: Partial or Full ?

SPFRI can be achieved using fully and a partially proactive flow rule injection.

6.2.1 Time Required to Install Flow Rules

Fully proactive flow rule injection installs all required flow rules to all switches in the service chain at once.However, if a partial proactive flow rule injection approach is followed, the corresponding switch must communicate with the controller at the beginning of each segment. Therefore, the time required to complete a flow rule injection is proportional to the number of segments i.e., the number of middleboxes in a particular service chain.
Time required to set up Service Chain = Time required to Map users traffic to Service ID using policy database + (Finding the paths between middleboxes and selection of path with respect to Service Chain ID + retrieving middlebox specific actions form middlebox database) * number of middleboxes in the service chain + maximum of communication between switch and controller.

6.2.2 Dynamic Policy Update

If the chaining policy is updated while fully proactively injecting flow rules, there may be inconsistency between the policy and its enforcement. As old rules from switches must be replaced with new rules, such time lag can be connected to a security threat or misbehaviour of traffic such as packet drops or unwanted traffic shaping. For partial proactive flow injection, such risks can be mitigated to some extent.

6.2.3 Total Number of Flow Rules

Assuming many to one mapping of users to the service chain, the number of flow rules is not proportional to the number of users. The number of flow rules per service chain at each switch along the path is two (one each for forward and reverse direction). With proactive flow rule injection, one additional flow rule must be inserted at the switch connected to a middlebox.
Total flow rules = (2 * no of service chains) * number of switches + no of middleboxes * no of service chains. Note that VLAN ID masking will reduce the number of flow rules.

6.2.4 Limitations

For each service chain, each middlebox in the sequence must configure a VLAN interface that corresponds to the service chain ID. In addition, the matching among VLAN ID, IP address, service and the middlebox must be maintained appropriately. This may slightly increase the operational cost of service chains.

Chapter 7

Evaluation

The ideal solution that implements service chaining should have following properties:

1. **No Modification to Middlebox :**
   The traffic must follow the sequence of middlebox traversal without modifying operating system or software of these middleboxes. The solution proposed NSH [6], FlowTag [2] fails as middleboxes need to be aware of tags. SPFRI requires no modification to operating system or software of middlebox.

2. **Mangling Middlebox :**
   Steering the traffic through desired sequence of middlebox even in the presence of middlebox that alters packet header is difficult to achieve as service chain context may get lost. The proposed solution uses VLAN ID field which remains unaltered as packet entering and existing through same VLAN interface. Thus, Service chain context remains intact even in the presence of mangling middlebox in the service chain.

3. **Friendliness with non SDN :**
   Steering [1] uses pipeline feature introduced with OpenFlow version 1.1. The service chain context is stored using 64-bits meta information which is not available in traditional switches and router. VLAN are very much popular in an enterprise network. Our proposed solution uses VLAN ID field to maintain the service chain context and works in case both traditional and SDN network.

4. **Scalability :**
   The solution should support longer and thousands of service chains with many middleboxes. Steering [1] supports only 63 service chain (1 bit is used for direction). Position [7] uses a service instance per user. With the number of users goes on increasing , flow rules to support increasing service instance increases. But SDN switches have limited capacity to support flow rules as TCAM has fixed and limited size. SPFRI support any length of service chain and any number of middleboxes.

5. **Less Computation Overhead :**
   Simple [3] sends multiple packets to the controller to correlate flow in the case of mangling middlebox that introduces additional overhead over a controller. For position [7], the service

instance deployment per user in the case of the small duration, flow rules introduce computation overhead as well as management of flow rules on SDN switches. NSH [6] and FlowTag [2] propose an additional header to carry service chain context that middlebox needs to be aware of. The middleboxes need to produce, process and consume these tag which an additional overhead.

6. **Less number of Flow Rules :**
   Position [7] requires per user a service instance. In SPFRI, many users are mapped to one service instance that will require one flow rule at ingress switch. Thus, saving more number of flow rules.

7. **Security :**
   OpenSCCaas [8] uses source MAC address to encode the service chain context. Thus we can not use MAC-based access control list.

The existing service chain solution approaches to steer the traffic through the desired sequence of middlebox fail to achieve at least one of the characteristic. Table 7.1 shows the comparison of SPFRI with the existing service chaining solutions.

Solution	NSH	StEERING	SIMPLE	FlowTag	Position	OpenSCaas	SPFRI
No Modification to Middlebox	**N**	**Y**	**Y**	**N**	**Y**	**Y**	**Y**
mangling Middlebox	**Y**	**N**	**Y**	**Y**	**-**	**Y**	**Y**
Friendliness with non SDN	**-**	**N**	**Y**	**Y**	**N**	**-**	**Y**
Scalability	**Y**	**N**	**-**	**-**	**N**	**-**	**Y**
Less Computation Overhead	**N**	**-**	**N**	**N**	**N**	**-**	**Y**
Less number of Flow Rules	**-**	**-**	**Y**	**-**	**N**	**-**	**Y**
Security	**-**	**-**	**-**	**-**	**-**	**N**	**Y**

Table 7.1: Comparison with the Existing approaches for Service Chaining

Chapter 8

Conclusion and Future Work

Service chaining is challenging because middleboxes modify the packet headers and the service context gets lost. While existing approaches suffer from critical drawbacks, we have designed and implemented a system that can guarantee correct middlebox traversal without any modifications to the middlebox software. The proposed SPFRI utilizes the VLAN ID in the VLAN header to store the contextual information for service chaining. Our specific contribution exploited existing technology with carefully managed and configured middleboxes to solve the service chaining problem. The proposed SPFRI is lightweight and works for any type of middlebox. We have theoretically verified that service chaining using SPFRI can be achieved even in the presence of altering middleboxes. We believe that the disadvantage of configuring VLAN interfaces on middleboxes is sufficiently negligible considering the significant benefit gained by service chaining with the proposed SPFRI, which provides longer and more service chains with consistent flow rule enforcement.

In future, a practical implementation of the proposed method will be a primary focus. Currently, 802.1AD is supported by OpenFlow 1.3 but is not widely available on open source software switches. However, Lagopus [15] appears promising for SPFRI using 802.1AD. We will practically implement the proposed SPFRI and explore its integration to the existing NFV software architecture for the purpose of real deployment and also further analysis of performance and operational overhead. We will be coming with some greedy algorithm to assign the tag to these service chain so that we will be able to reduce the number of flow rule on switches.

References

[1] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan *et al.*, “Steering: A software-defined networking for inline service chaining,” in *Network Protocols (ICNP), 2013 21st IEEE International Conference on.* IEEE, 2013, pp. 1–10.

[2] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, “Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags,” in *Proc. USENIX NSDI*, 2014.

[3] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, “Simple-fying middlebox policy enforcement using SDN,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[5] IEEE 802.1AD, IEEE Std. 802.1AD, 2005. [Online]. Available: http://standards.ieee.org/findstds/standard/802.1ad-2005.html

[6] P. Quinn, R. Fernando, J. Guichard, S. Kumar, P. Agarwal, R. Manur, A. Chauhan, M. Smith, N. Yadav, B. McConnell, and C. Wright., “Network service header,” *Internet-Draft draft-quinn-nsh-03, IETF Secre-tariat*, July 2014.

[7] J. J. Blendin, J. Ruckert, N. Leymann, G. Schyguda, and D. Hausheer, “Position: Software-defined network service chaining,” in *European Workshop on Software Defined Network (EWSDN).* IEEE, 2014, pp. 139 – 140.

[8] J. W. Wanfu Ding, Wen Qi and B. Chen, “Openscaas: an open service chain as a service platform toward the integration of sdn and nfv,” in *Network, IEEE*, vol. 29. IEEE, May-June 2015.

[9] The frenetic project, 2013. [Online]. Available: http://frenetic-lang.org

[10] A. B. B. Lopes and M. P. Fernandez, “Ponderflow: A policy specification language for openflow networks,” in *Proceedings of ICN : The Thirteenth International Conference on Networks*, vol. 13, 2014.

[11] K. Twidle, N. Dulay, E. Lupu, and M. Sloman, “Ponder2: A policy system for autonomous pervasive environments,” in *2009 Fifth International Conference on Autonomic and Autonomous Systems*, April 2009, pp. 330–335.

[12] Floodlight. [Online]. Available: http://www.projectfloodlight.org

[13] IEEE 802.1Q, IEEE Std. 802.1Q, 2005. [Online]. Available: http://www.ieee802.org/1/pages/802.1Q.html

[14] A. V. E. Rosen and R. CallIon, “Multiprotocol label switching architecture,” Internet Engineering Task Force, 2001, rFC 3031.

[15] Lagopus. [Online]. Available: https://lagopus.github.io/