Chapter 7

Waypoint Enforcement in Hybrid SDN

7.1 Introduction

In previous chapter we surveyed various works done in Hybrid SDN. There are many solutions in literature for incremental deployment of SDN enabled devices in legacy networks [222, 65, 68, 72]. The deployment of SDN switches in the existing network alone cannot help much. In a network with SDN switches, it must be ensured that the network traffic should go through SDN switches to leverage the power of SDN. So the network traffic needs to be diverted from its original path to the SDN switch. We call this *"waypoint enforcement"*. Waypoint enforcement is defined as constraining the network traffic to take the path with at least one SDN switch. Once a packet reaches the SDN switch, it

- Sandhya, Mallikarjun Swamy, K. Haribabu, and Ashutosh Bhatia. *Achieving waypoint enforcement in multi-VLAN Hybrid SDN*, In 10th International Conference on Communication Systems & Networks (COMSNETS), pp. 519-521. IEEE, 2018.
- Sandhya Rathee, T Dinesh Ram Kumar, K Haribabu, and Ashutosh Bhatia. *International Conference on Advanced Information Networking and Applications (AINA-2019)*, pp. 325-340. Springer, Cham, 2019.
- Sandhya Rathee, K. Haribabu, Parth Patel, Ashutosh Bhatia, and Ujjwal Gandhi. *Analysis and Performance Evaluation of Different Methods to Achieve Waypoint Enforcement in Hybrid SDN*, In Workshops of the International Conference on Advanced Information Networking and Applications (WAINA-2020), pp. 190-200, Springer, Cham, 2020.

is subjected to the policies installed by the network controller. Thus, waypoint enforcement would help us realise real-time network-wide policy enforcement [166], consistent network policy updates [223], etc.

In literature, there are a few works [68, 66, 67] which provide different solutions to achieve waypoint enforcement. Panopticon [68] divides the entire network into cell blocks by deploying a few SDN devices in the existing network. It designates some ports of legacy devices as SDN_c ports and others as legacy ports. Panopticon [68] achieves full waypoint enforcement for inter-cell traffic and the traffic going from or coming to an SDN_c port. It does not provide control over intra-cell traffic, which has source and destination as legacy ports. Thus, it provides partial waypoint enforcement. Telekinesis [66] and Magento [67] are probabilistic solutions for layer-2 networks. They require at least one SDN switch to be present in each subnet to enforce waypoint enforcement. They also provide partial waypoint enforcement.

In this chapter, we propose two frameworks for waypoint enforcement. First, we propose a novel framework to achieve full waypoint enforcement even with a single SDN switch present in the network. It uses unused IP addresses as virtual IP addresses to achieve full waypoint enforcement. When a source host wants to communicate with another host, the source is provided with the virtual IP address of the destination. All packets destined for a virtual IP address are routed to an SDN switch. The SDN switch then routes the packets to the actual destination. Thus, it achieves full way point enforcement. The second framework overcomes the limitation of Magneto [67] and Telekinesis [66]. It does not require an SDN switch per subnet to achieve waypoint enforcement. Unlike Magneto, it uses a multi-homed host to send gratuitous Address Resolution Protocol (gARP) packets to poison the Address Resolution Protocol (ARP) table of every host, such that every packet from a given host is diverted towards the nearest SDN switch. However, it provides only partial waypoint enforcement.

7.2 Problem Formulation

In this section, we describe the problem of achieving waypoint enforcement in Hybrid enterprise SDN networks. We consider a three-tier architecture given by Cisco [135]. The

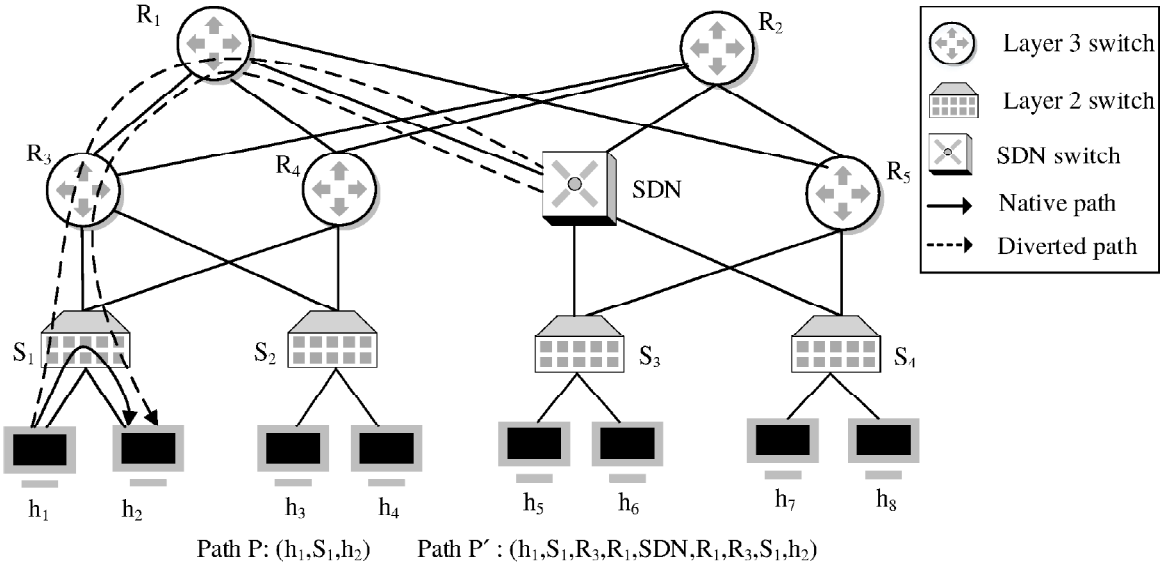

Figure 7.1: Waypoint enforcement in Hybrid SDN network.

network consists of three layers, namely core, distribution, and access layer. All the hosts are directly connected to access layer switches. To achieve SDN control in Hybrid SDN networks, where only a few switches are SDN enabled, traffic must flow through at least one SDN switch. Table 7.1 lists the symbols used in this section.

Definition 1 (**Native Path**). The path between any two nodes in a network is called the Native Path.

Definition 2 (**Diverted Path**). For a given pair of source $h_{i,j}^{k,l}$ and destination $h_{i',j'}^{k',l'}$ host, diverted path is the path which consists of at least one SDN switch[1]

For example, given a network as shown in Figure 7.1. Let host h_1 communicate with host h_2. The native path between h_1 and h_2 is through access switch S_1. Let a few SDN switches be deployed at the distribution layer of the network, as shown in Figure 7.1. To achieve waypoint enforcement, the $h_1 - h_2$ communication traffic is diverted to pass through at least one SDN switch. This results in path divergence and gives the diverted path as $h_1, S_1, R_3, R_1, SDN, R_1, R_3, S_1, h_2$.

Definition 3 (*Diversion point*). Let N_P($h_{i,j}^{k,l}$, $h_{i',j'}^{k',l'}$) be the native path between $h_{i,j}^{k,l}$ and $h_{i',j'}^{k',l'}$ and $SDN_{i''}^{j''}$ be the SDN switch. Diversion point is the point in the native path at which the traffic detours its path to include an SDN switch. The diversion point can be an access switch, distribution switch, or a core switch in the native path, N_P($h_{i,j}^{k,l}$, $h_{i',j'}^{k',l'}$). For

[1]A native path can itself include one or more SDN switches. In such a case, we do not need a diverted path as the traffic is already going through SDN switch/es.

example, in Figure 7.1 switch S_1 is the diversion point for communication between host h_1 and host h_2.

Table 7.1: Summary of used Variables.

Variable	Description
C_i	Indicates the i^{th} core switch in the network.
D_i^j	Indicates a legacy distribution switch, such that D_i^j is j^{th} distribution switch of i^{th} core switch
$A_{i,j}^k$	Indicates an access switch, such that $A_{i,j}^k$ is k^{th} access switch of D_i^j distribution switch
$h_{i,j}^{k,l}$	Indicates a host machine l connected to $A_{i,j}^k$
SDN_i^j	Indicates an SDN switch at distribution layer, such that SDN_i^j is the j^{th} SDN distribution switch of i^{th} core switch
D_{pt}	Indicates diversion point
$N\_P(N_1, N_2)$	Indicates a native path between node N_1 and node N_2 in the network
$D\_P(N_1, N_2)$	Indicates a diverted path between node N_1 and node N_2 in the network
$EP(N_1, N_2)$	Indicates extended path between node N_1 and node N_2 in the network

Given a communication pair ($h_{i,j}^{k,l}$, $h_{i',j'}^{k',l'}$) and an SDN switch, $SDN_{i''}^{j''}$, a diversion point can be identified using Algorithm 7.1. If both the hosts $h_{i,j}^{k,l}$ and $h_{i',j'}^{k',l'}$ are connected to the same access switch, then the divergence point is the access switch only, i.e., $A_{i,j}^k$ (line 1-2 of Algorithm 7.1 on the following page). If both the hosts $h_{i,j}^{k,l}$ and $h_{i',j'}^{k',l'}$ are not connected to the same access switch, then check the distribution switch/es. If the source host $h_{i,j}^{k,l}$ and destination host $h_{i,j}^{k,l}$ are under the same distribution switch and the distribution switch is not an SDN switch, then the distribution switch D_i^j is the diversion point (line 4-5 of Algorithm 7.1 on the next page). If both the hosts $h_{i,j}^{k,l}$ and $h_{i',j'}^{k',l'}$ are not under the same distribution switch, and neither of the distribution switches is an SDN switch, then check the core switch/es (line 6 of algorithm 7.1 on the following page). If the SDN switch is under i^{th} core switch, then the diversion point is the core switch C_i (line 7-8 of algorithm 7.1 on the next page). Else $C_{i'}$ is the diversion point (line 9-10 of algorithm 7.1 on the following page).

$D\_P(h_{i,j}^{k,l}, h_{i',j'}^{k',l})$ always have a diversion point and the diverted path can be calculated

Algorithm 7.1: Algorithm to identify the diversion point

Input : Communication pair ($h_{i,j}^{k,l}$, $h_{i',j'}^{k',l'}$)
Output : Diversion point

if ($k == k'$) **then**
  // both the hosts are connected to same access switch
  $D_{pt} \leftarrow A_{i,j}^{k}$; // the access switch is divergence point
else
  // both the hosts are not connected to same access switch
  if ($j == j'$ *and* $j \neq j''$) **then**
    // source and destination hosts are under same distribution switch which is not an SDN switch
    $D_{pt} = D_{i}^{j}$
  else if ($j \neq j''$ *and* $j' \neq j''$) **then**
    // source and destination hosts are not under same distribution switch and neither of the distribution switches is an SDN switch
    if ($i == i''$) **then**
      // source host's core switch has a connection to the SDN switch
      $D_{pt} = C_{i}$;
    else
      // source host's core switch has a connection to the SDN switch
      $D_{pt} = C_{i'}$;
    end
  end
end

as follows,

$$P_1 = N\_P(h_{i,j}^{k,l}, D_{pt}) \tag{7.1}$$

$$P_2 = N\_P(D_{pt}, SDN_{i''}^{j''}) \tag{7.2}$$

$$P_3 = N\_P(SDN_{i''}^{j''}, D_{pt}) \tag{7.3}$$

$$P_4 = N\_P(D_{pt}, h_{i',j'}^{k',l'}) \tag{7.4}$$

since, $|$ N_P(D_{pt}, $SDN_{i''}^{j''}$)$|$ = $|$N_P($SDN_{i''}^{j''}$, D_{pt})$|$. So,

$$|P_2| = |P_3| \tag{7.5}$$

$$D\_P(h_{i,j}^{k,l}, h_{i',j'}^{k',l'}) = |P_1| + 2|P_2| + |P_4| \tag{7.6}$$

Definition 4 (*Extension*). The extension in the path is defined as the difference between the length of the native path and the length of the corresponding diverted path. That is,

$$|EP(h_{i,j}^{k,l}, h_{i',j'}^{k',l'})| = |D\_P(h_{i,j}^{k,l}, h_{i',j'}^{k',l'})| - |N\_P(h_{i,j}^{k,l}, h_{i',j'}^{k',l'})| \tag{7.7}$$

7.3 Methods To Achieve Waypoint Enforcement

In literature, there exist many solutions for incremental deployment of SDN switches in the existing network. However, only a few of them [224, 67, 68, 66, 225] focus on achieving waypoint enforcement in Hybrid SDN networks. In this section, we explain existing methods to achieve waypoint enforcement.

7.3.1 Panopticon

Panopticon [68] proposes a mechanism to achieve waypoint enforcement in Hybrid SDN enterprise networks. The whole network is divided into multiple cell blocks. A cell block is defined as the set of the connected legacy nodes obtained after removing all the SDN switches and the links incident on them. The adjacent SDN switch to a cell block is called the frontier of the cell block. The ports of all the legacy switches in the network are divided into two sets, (1) the ports which need to be exposed to and controlled by the SDN controller (called SDN_c ports) and (2) the ports which are normal ports (called legacy ports). The main idea is that the traffic which has SDN_c port as source or destination has to go through the SDN switch. To achieve this, panopticon [68] builds a solitary confinement tree (SCT) from every SDN_c port in a given cell block to the frontier of that cell block (i.e., adjacent SDN switch of the cell block). SCT is a spanning tree of cell block plus the SDN switch/es. The root of every SCT in a given cell block is an SDN_c port. The primary role of SCT is to determine a path from a given SDN_c port to an SDN switch. To ensure traffic isolation, VLAN ID is assigned to every SCT. Thus, panopticon provides a logical SDN view of the Hybrid SDN network to the SDN controller. Consider a network with seven nodes (including both legacy and SDN switches) as shown in Figure 7.2 (a). There are four cell blocks in the network, with a total of six SDN_c ports and two SDN switches. SDN controller's view is limited only to the SDN_c ports and SDN switches, as shown in Figure

7.2 (b). Continuing the same example as shown in Figure 7.2 (a), we illustrate the working

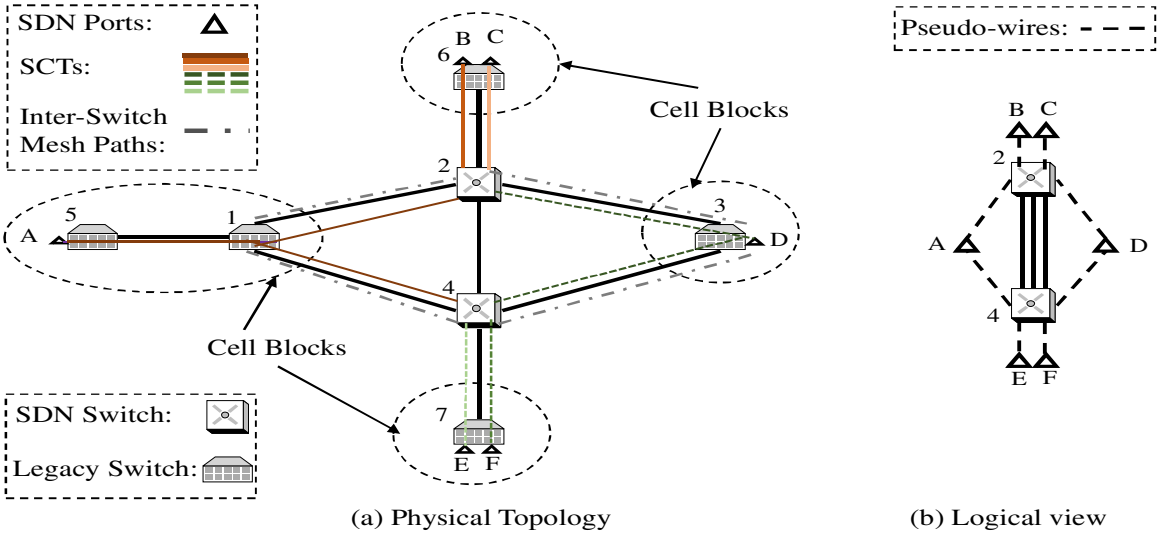

Figure 7.2: Example of Hybrid SDN network with seven switches (including both SDN and legacy). (a) Physical network with solitary Confinement Trees (SCTs) for each SDN_c port. (b) The logical view of SDN controller.

of Panopticon. The communications can be divided into the following categories,

1. When either source or destination or both for a given communication is/are connected to SDN_c port/s. For example, consider the traffic between SDN_c port x and y. When packets from x enter SCT of x (i.e., SCT(x)), a VLAN tag of that SCT's VLAN is added to the packets, and the packets are forwarded towards the frontier of x (i.e., SDN switch). Now there are two cases, one where the SDN switch acts as a VLAN gateway. This case occurs when the cell block of x and the cell block of y shares a common frontier. For example, node 2 (i.e., SDN switch) in Figure 7.2 (a) acts as a gateway for the communications between SDN_c port A and SDN_c ports B, C, and D. In this case, the gateway SDN switch replaces the SCT(x)'s VLAN tag with SCT(y)'s VLAN tag before forwarding the packet to y's cell block. In the second case, when the source and destination cell blocks do not share a frontier, a tunnel is created between the frontier of SCT(x) and the frontier of SCT(y). For example, in Figure 7.2 (a), for communications between B or C and E or F, a tunnel is created between node 2 and node 4.

2. When neither the source nor the destination of a given communication is SDN_c port, the packet forwarding is done by the legacy switches and is not affected by the SDN

policies.

In other words, panopticon can only control the traffic, which has an SDN_c port as either source or destination or inter-cell traffic. It would not control the intra-cell traffic, which has source and destination as legacy ports. Thus, it does not guarantee full or 100% traffic waypoint enforcement. Panopticon is highly dependent on VLAN IDs. Thus it can disturb the existing VLAN configurations as well.

7.3.2 Telekinesis

Telekinesis [66] proposes a framework to provide fine-grained control over forwarding in layer 2 switches in Hybrid SDN networks. The authors introduce a new control primitive called *"LegacyFlowMod"*. The controller uses *LegacyFlowMod* primitive to instruct the SDN switch/es to send a special packet to the legacy switches. When a special packet reaches a legacy switch, it updates the forwarding entry of the legacy switch.

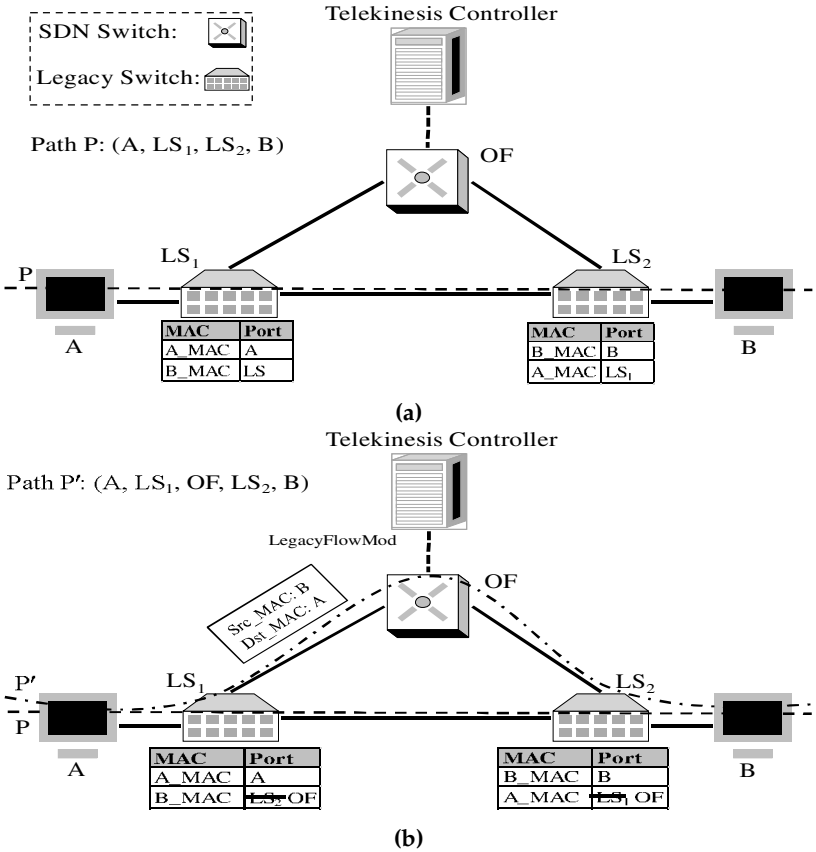

Figure 7.3: An example of path update in Hybrid SDN network using Telekinesis. (a) Native path from source A to destination B. (b) Change in path from source A to destination B through SDN switch.

Consider the network given in Figure 7.3 (a). The path from host A to host B is A, LS_1, LS_2, B. However, to achieve waypoint enforcement the path should be A, LS_1, OF, LS_2, B. The SDN controller calls the *LegacyFlowMod* to send a special packet from OF (OpenFlow) switch to update the forwarding entry of LS_1 switch, for the communication between host A and host B, such that, the traffic takes the path which includes the OF switch. The source MAC address of the special packet is that of host B, and the destination MAC address is that of host A. When this special packet reaches LS_1 switch, it updates its forwarding table for host B from LS_2 switch to OF switch, as shown in Figure 7.3 (b). Now the traffic from host A to host B is forwarded through the updated path which is A, LS_1, OF, LS_2, B. When the traffic from host A to host B reaches LS_2 switch through OF switch, it updates its forwarding table for host A from LS_1 switch to OF switch. However, after updating the forwarding entry of LS_1 switch, for the communication between host A and host B, it is possible that a packet from destination host B arrives at LS_1 switch from LS_2 switch. This can revert the forwarding entry at LS_1 switch from OF to LS_2 switch. Even if the controller sends special packets to both the source and the destination switches simultaneously, a packet in transit on the data channel can revert the forwarding entry of LS_1 switch to the original one. Thus, the controller has to send the special packets at least at the rate of the traffic to stabilize the forwarding entries of the legacy switches.

To divert the traffic to go through an SDN switch, the special packet should reach all the legacy switches which constitute the new path. For example, consider a network given in Figure 7.4(a) and a path in the network, say P_1. To update the path for MAC address x from P_1 to $P_1^{'}$, the special packet with source MAC address x should reach LS_2 switch at port 3 and LS_3 switch at port 2 from the OF switch. When LS_2 switch receives the special packet at port 3, it assumes that the MAC address x is reachable through port 3 and updates its forwarding entry. Similarly, the forwarding entry of LS_3 switch is updated for MAC address x with port 2. Thus, path P_1 gets updated to path $P_1^{'}$.

Telekinesis needs at least one SDN switch to be connected to any of the legacy switches of the original path. For example, consider the network given in Figure 7.4(a), there are two paths in the network P_1 and P_2. The path P_1 is updated as explained above whereas, the path P_2 cannot be updated because no legacy switch of path P_2 is connected to an SDN

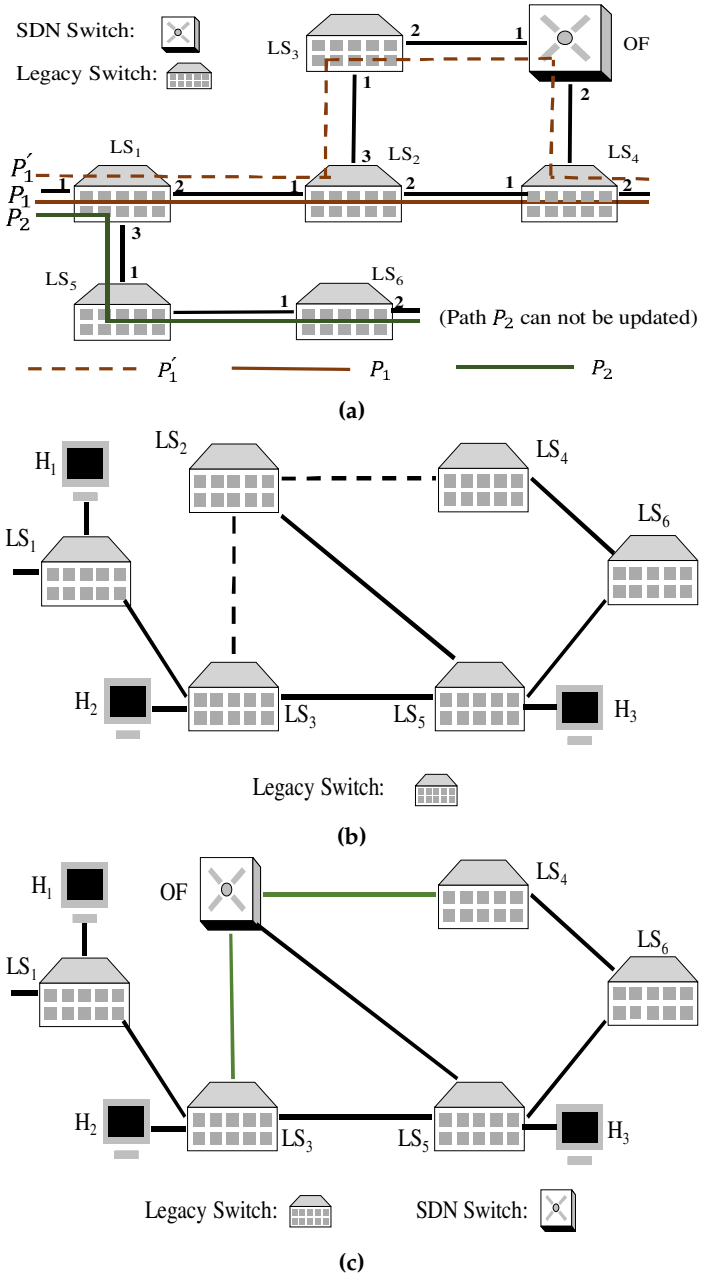

Figure 7.4: Examples to illustrate the limitations of Telekinesis. (a)Path Updates (b) Legacy network with communication pairs H_1-H_3 and H_2-H_3 (c) Effect of updating in MAC learning table of switch LS_3.

switch. Another limitation of Telekinesis is that updating a path for a given destination updates all the paths for that destination. For example, consider a legacy network given in Figure 7.4(b), using Spanning Tree Protocol (STP), we can avoid loops in the network. After running STP, the links which are not part of the network topology are shown with dashed lines. Consider, host H_1 and H_2 both are sending traffic to host H_3. Now assume that switch LS_2 is replaced with an OF switch, as shown in Figure 7.4(c). Now the links

from switch LS_3 to OF switch and from OF switch to LS_4 switch can be used to achieve waypoint enforcement. The OF switch sends a special packet to LS_3 switch with source MAC address of host H_3. This updates the forwarding entry at switch LS_3. Now, LS_3 switch sends all the packets destined for host H_3 to OF switch. Thus, it updates the path for both the communications (i.e., from host H_1 to host H_3 and from host H_2 to host H_3).

7.3.3 Magneto

Magneto [67] is an enhancement over Telekinesis. They extend the telekinesis framework to exert control over the legacy devices using SDN switches. They assign a magnet MAC address to the path for a destination. The magnet MAC address is not part of the network. The difference is that the source MAC address of the special packet is the magnet MAC address instead of the destination's actual MAC address. Another difference is that the controller instructs the SDN switch/es to send an ARP packet/s instead of a special packet. The ARP packet updates the ARP table of the source host as well as the forwarding table of the intermediate switches. For example, consider the network given in Figure 7.5, to update the path from source A to destination B from (A, LS_1, LS_2, B) to (A, LS_1, OF, LS_2, B). The controller sends a unicast ARP packet with source MAC address as B_MAC' (magnet MAC address) to host A. While traversing the path to host A from OF switch, the packet also updates the forwarding table of switch LS_1 for B_MAC'. When the ARP packet reaches host A, it adds an IP to MAC entry in its ARP table for host B. Similarly, the OF switch updates the forwarding table of LS_2 switch and host B's ARP table for host A.

Magneto does not update all the paths for a given destination. For different source hosts, the controller can assign different magnet MAC addresses to a given destination. Using these MAC addresses, the controller can update paths for a destination individually to diverge traffic through OF switch. Thus, it overcomes the limitation of Telekinesis. As explained already in Section 7.3.2, the path update for the communication pair H_1-H_3 in Figure 7.4 (b) and 7.4 (c) also updates the path for H_2-H_3 communications. Whereas, in Magneto, the controller can use different magnet MAC addresses for host H_3 w.r.t to the source hosts H_1 and H_2.

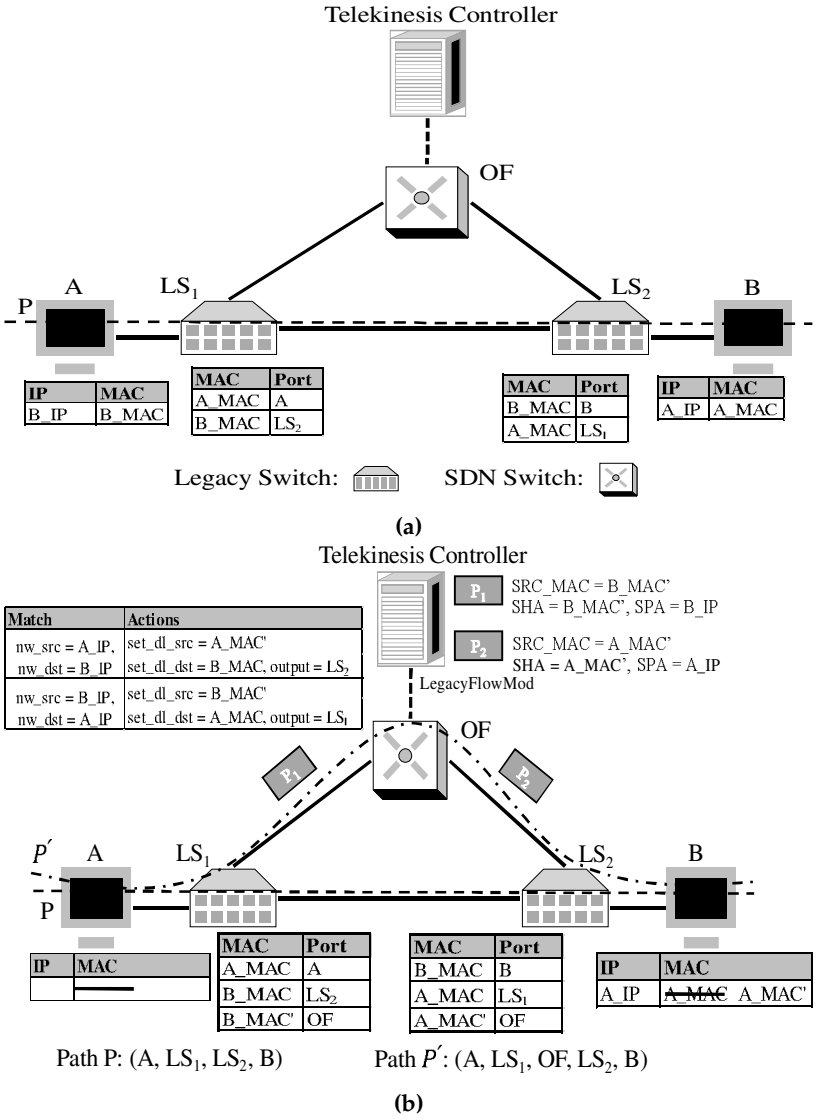

Figure 7.5: An example of path update in Hybrid SDN network using Magneto framework. (a) Native path from source A to destination B. (b) Change in path from source A to destination B through SDN switch.

Magneto also overcomes the limitation of the path flipping problem in Telekinesis. For example, if the controller is updating the path from source host A to destination host B given in Figure 7.5. The OF switch sends an ARP packet with magnet MAC address B_MAC' to host A. While traversing the path to host A, the LS_1 switch learns the magnet MAC address of host B (i.e., B_MAC') and adds an IP to MAC mapping in host A's ARP table. Even if there is any packet from host B to A with native MAC address of host B, it does not cause path flipping because it will be forwarded based on native MAC address. Once the controller updates the ARP table of host B for source A, the traffic starts going through the OF switch. Both Telekinesis and Magneto require an SDN switch in each

subnet to achieve waypoint enforcement.

7.4 A Method for Enhancing Magneto

Telekinesis [66] and Magneto [67] are limited to a subnet. They require at least one SDN switch to be present in each subnet to guarantee waypoint enforcement. To solve this issue we propose a solution to achieve waypoint enforcement by deploying as few as one SDN switch in the network. The proposed solution uses gratuitous ARP (gARP) packets [226] to poison the ARP cache of every host, such that the traffic is diverted towards the nearest SDN switch. We use a multi-homed host machine (called server), which is part of every VLAN, instead of an SDN switch to poison the ARP cache. Since the server is part of every VLAN, it can send a broadcast gARP packet in all VLAN such that each packet go through an SDN switch. Thus, it overcomes the limitation of Telekinesis [66] and Magneto [67].

For example consider the network given in Figure 7.6 and hosts h_1 and h_2 are communicating with each other. The native path, P, from host h_1 to host h_2 is h_1, S_1, S_2, h_2 (as shown in Figure 7.6 (a). The SDN controller supervises the server and instruct it to update the communication path from hosts h_1 and h_2 through SDN switch (i.e., OF). To divert the traffic towards the SDN switches, the server broadcasts gARP packet P_1 with source MAC address as h_2_MAC' as shown in Figure 7.6 (b). When the gARP packet reaches host h_1, it adds an IP to MAC entry in the ARP cache for host h_2. While traversing the path to hosts in the network, the gARP packets also update the forwarding cache of switches for h_2_MAC'. Similarly the server updates the forwarding tables of switches and host h_2's ARP cache for host h_1. When the packets from host h_1 reach the OF switch, it forwards the packets to the intended destination and the diverted path, $P^{'}$, from host h_1 to h_2 is h_1, S_1, OF, S_2, h_2.

7.4.1 Performance Evaluation

Traffic is generated between two adjacent hosts to determine the efficiency of the proposed mechanism. To perform the experiments, we have used Mininet [137]. The performance of the proposed mechanism is measured using following two parameters:

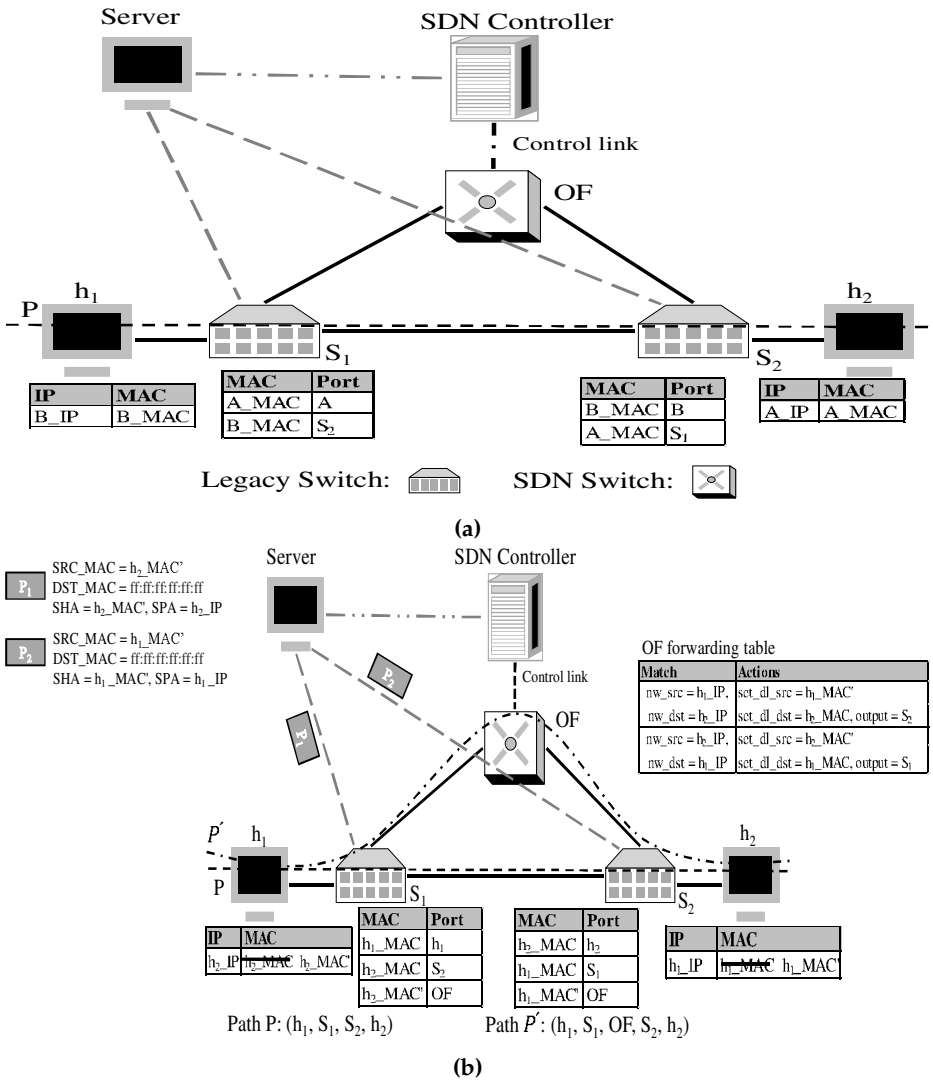

Figure 7.6: An example of path update in Hybrid SDN network. (a) Native path from source A to destination B. (b) Change in path from source A to destination B through SDN switch.

1. **Percentage of waypoint enforcement achieved:** It is defined as the ratio of number of packets reaching SDN switch to the total number of packets generated.

2. **Control overhead:** It is defined as the bandwidth used by the gARP packets sent by the server to poison the ARP cache of host machines.

Figure 7.7 indicates waypoint enforcement performance w.r.t varying waiting time period. The waiting time period refers to the delay between every successive gARP broadcast cycle. The reason for decrease in waypoint enforcement ratio w.r.t increase in waiting time period (i.e., decrease in the frequency of sending the gARP packets) is due to the fact that, by the time the host receives a fresh gARP packet, it would have already discarded

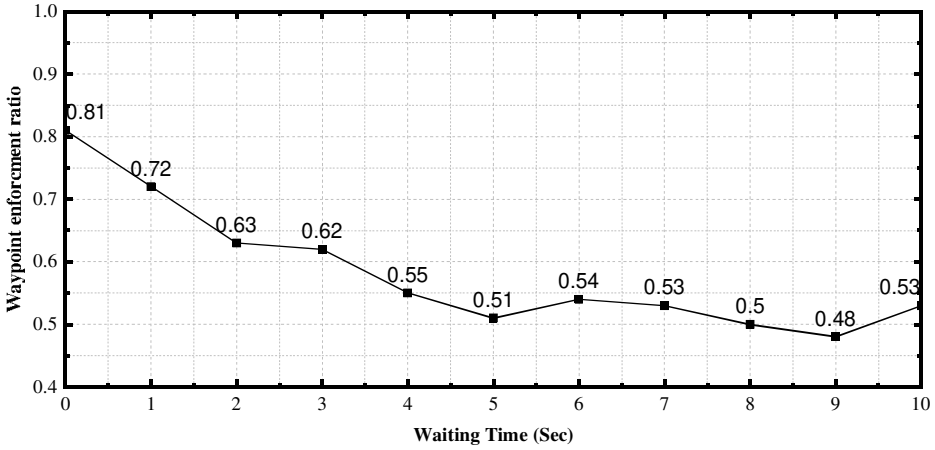

Figure 7.7: Percentage of packets reaching to SDN switch.

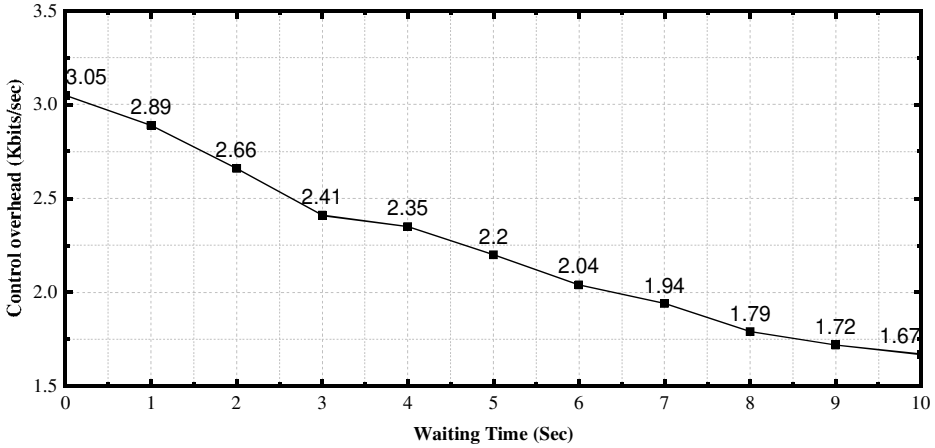

Figure 7.8: Overhead caused by the gARP packets

the poisoned ARP entries and populated the ARP cache by initializing the ARP protocol.

ARP entry is removed from the ARP cache if it is not used for a specified amount of time called "ARP cache timeout". As a result, the packets transmitted by a host to a particular destination during the period after the ARP cache entry for the destination has been removed and before the reception of the next gARP packet, would not be directed towards the SDN switch and take their native path to reach the destination. This suggests that, to increase the degree of waypoint enforcement we have to increase the frequency of ARP poisoning so that the hosts find poisoned ARP entry in the ARP cache majority of the time. However, an increase in the frequency of ARP poisoning will also increase the overhead in the network as it is evident from Figure 7.8. Therefore, the degree of waypoint enforcement that can be achieved in a network using the proposed method can be traded-off with the overhead incurred due to the frequent poisoning of ARP cache.

7.5 Waypoint Enforcement Using Virtual IP

The goal is to achieve waypoint enforcement by deploying as few as one SDN switch in the network. In this section, we discuss two frameworks to achieve waypoint enforcement in enterprise networks. The objective of the first framework is to achieve full waypoint enforcement. The state-of-the-art methods discussed earlier achieve only partial waypoint enforcement. The second framework overcomes the limitation of telekinesis and Magneto. Both the earlier works require at least one SDN switch in every subnet to achieve waypoint enforcement. Whereas the proposed framework does away with this requirement and still achieves waypoint enforcement.

We take a set of IP addresses that are not assigned to any host in the network, called *virtual IP addresses*. For every host in the network, the controller maps a virtual IP address with their real IP address. Before any communication starts in the network, the source hosts are required to send a DNS (Domain Name Server) query to resolve the destination host's IP address. DNS server maintains the mapping of the real IP address of the host to its virtual IP address. When a DNS query reaches the DNS server, it replies with the destination host's virtual IP address instead of its real IP address. The SDN controller supervises the DNS server. The mapping between virtual IP addresses and real IP addresses is known only to the SDN switch/es. For all virtual IP addresses, SDN switch acts as a gateway. The source host creates a packet with a virtual IP address as the destination IP address, which gets diverted towards the SDN switch. Once a packet reaches the SDN switch, it is forwarded to the intended destination using its real IP address. To achieve this, the SDN controller adds the flow entries in the SDN switches. The flow entries have actions to modify the packet's destination virtual IP address with its corresponding real IP address and its source real IP address with the corresponding virtual IP address. Thus, the destination host receives the packet with the virtual IP address of the source host. It responds to source host using this virtual IP address. The same procedure applies when the destination replies.

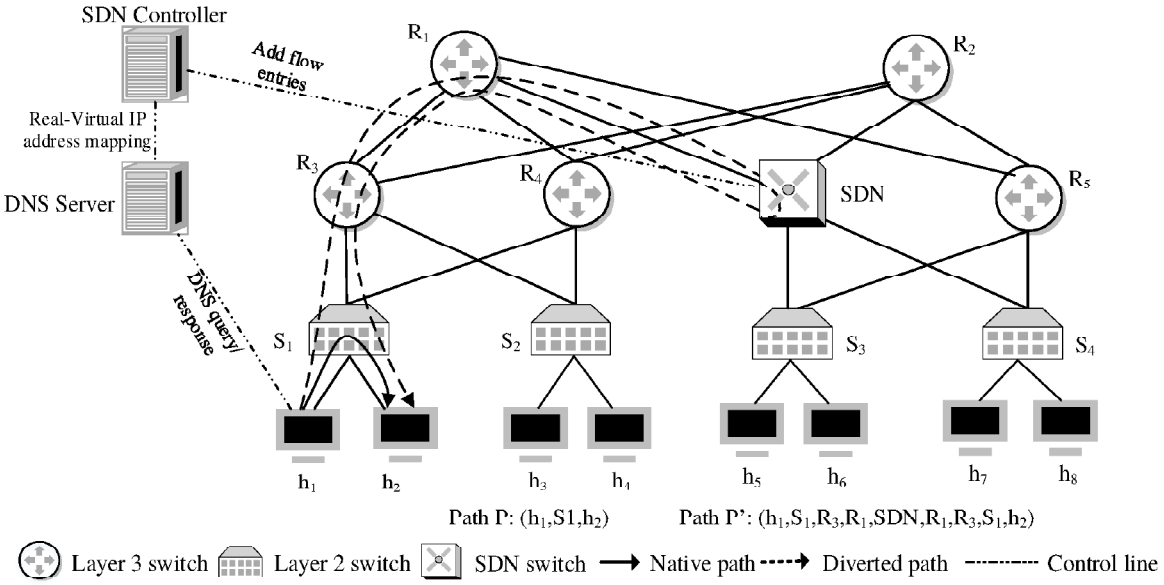

Figure 7.9: Path setup in Hybrid SDN network.

7.5.1 Virtual IP Address Mapping

In this section, we discuss different models to allocate virtual IP addresses to the hosts in the network and show how inter-subnet and intra-subnet traffic is forwarded towards an SDN switch. All the virtual IP addresses are accessible via SDN switch, as the SDN switch acts as a gateway to the virtual IP addresses. We use GNS3 (Graphical Network Simulator-3) [227] to emulate the proposed framework. GNS3 allows to design complex networks and emulate them by configuring the devices ranging from simple workstation machines to powerful Cisco routers. We use the topology given in Figure 7.10 for emulations, which consists of three subnets 10.0.0.0/24 (Subnet 1), 10.0.1.0/24 (Subnet 2), and 10.0.2.0/24 (Subnet 3). We use Cisco 2691 [228] switches at the core layer and distribution layer, which contains $R1$, $R2$, $R3$, and $R4$ as shown in Figure 7.10. Cisco IOSvL2 switch is used at the access layer which contains S_1, S_2, and S_3.

7.5.1.1 One-to-One Mapping

The simplest mapping method is to use a one-to-one mapping between the host's *real IP address* and *virtual IP address*. In this model, we have two domains called *real IP address domain* and *virtual IP address domain*, and each domain has the same number of IP addresses. The mapping is a one-to-one function from *real IP address* to *virtual IP address*.

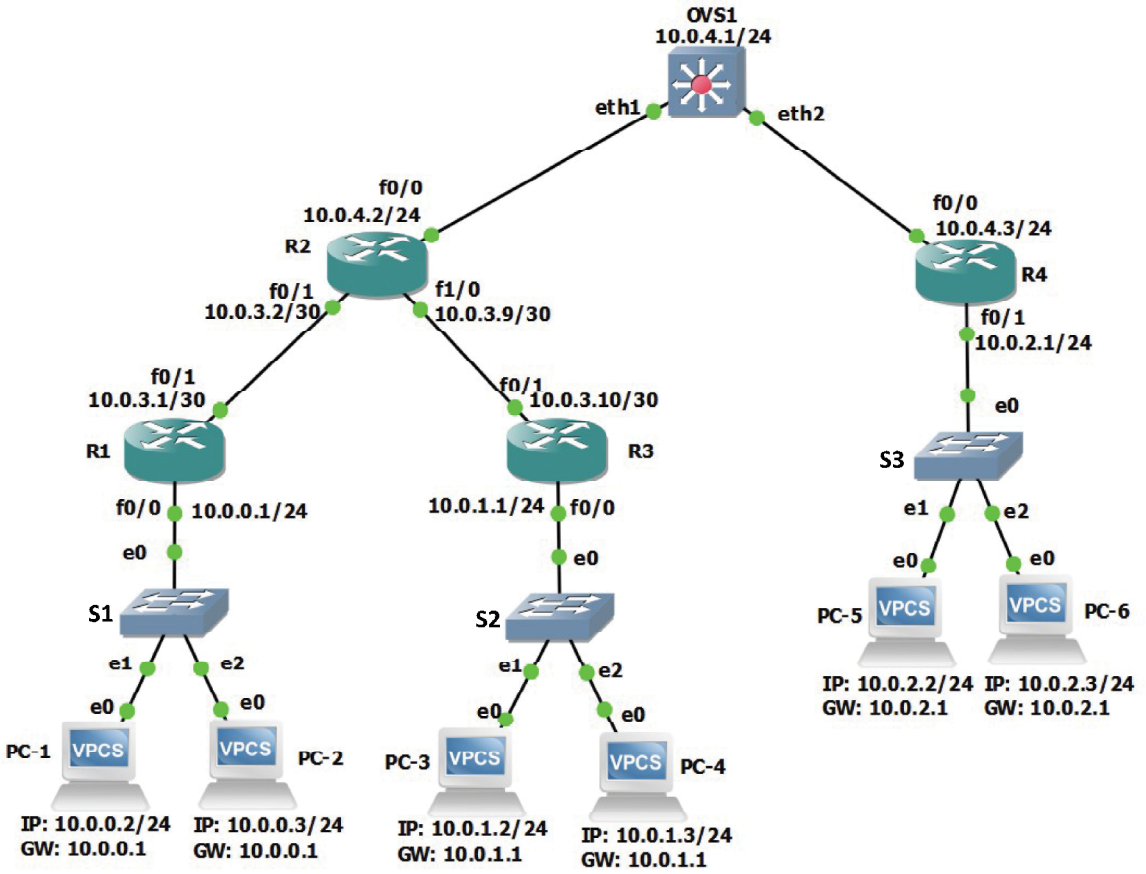

Figure 7.10: Topology for GNS3 emulations.

We consider three virtual IP subnets, one for each of the host subnet given in Figure 7.10. Let the virtual subnets be 172.16.0.0/24 (Virtual subnet 1), 172.16.1.0/24 (Virtual Subnet 2), and 172.16.2.0/24 (Virtual Subnet 3). Table 7.2 shows One-to-One mapping of real IP addresses to virtual IP addresses.

Table 7.2: One-to-One mapping

REAL IP	VIRTUAL IP
10.0.0.0/24	172.16.0.0/24
10.0.1.0/24	172.16.1.0/24
10.0.2.0/24	172.16.2.0/24

Now we need to configure the network to handle these virtual IP addresses. Whenever any legacy device encounters a packet with a destination address in the virtual IP domain, it must forward the packet towards the SDN switch. In the topology given in Figure 7.10, R1 and R3 must forward the packet to R2. R2 and R4 must forward the packet to OVS1. Thus, we define routes in R1, R2, R3, and R4 routers as given in Figure 7.11 and Figure

7.12.

R1 Configurations	R2 Configurations
R1#configure terminal	R2#configure terminal
R1(config)#ip route 172.16.0.0 255.255.255.0 10.0.3.2	R2(config)#ip route 172.16.0.0 255.255.255.0 10.0.4.1
R1(config)#ip route 172.16.1.0 255.255.255.0 10.0.3.2	R2(config)#ip route 172.16.1.0 255.255.255.0 10.0.4.1
R1(config)#ip route 172.16.2.0 255.255.255.0 10.0.3.2	R2(config)#ip route 172.16.2.0 255.255.255.0 10.0.4.1

Figure 7.11: Router R_1 and R_2 configuration for one-to-one mapping model.

R3 Configurations	R4 Configurations
R3#configure terminal	R4#configure terminal
R3(config)#ip route 172.16.0.0 255.255.255.0 10.0.3.9	R4(config)#ip route 172.16.0.0 255.255.255.0 10.0.4.1
R3(config)#ip route 172.16.1.0 255.255.255.0 10.0.3.9	R4(config)#ip route 172.16.1.0 255.255.255.0 10.0.4.1
R3(config)#ip route 172.16.2.0 255.255.255.0 10.0.3.9	R4(config)#ip route 172.16.2.0 255.255.255.0 10.0.4.1

Figure 7.12: Router R_3 and R_4 configuration for one-to-one mapping model.

The configuration of OVS1 is given in Figure 7.13. OVS1 checks if the destination IP address belongs to the virtual IP address space. If not, the packet matches the first flow entry, which has an action to process the packets as a normal L2/L3 device. If yes, it modifies the packet's source MAC to that of OVS1, and submits the packet to table 1 for further processing. Table 1 modifies the source IP address to the corresponding virtual IP address, and destination IP address to the corresponding real IP address and sends the packet to table 2 to decide where to send the resultant packet. In table 2, based on the destination IP address, it decides the next hop of the packet and modify the destination MAC address accordingly.

OVS1 Configuration
OVS1 # ovs-ofctl add-flow br0 table=0,priority=0,actions=NORMAL
OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=172.16.0.0/24,action=move:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[],resubmit(,1)"
OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=172.16.1.0/24,action=move:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[],resubmit(,1)"
OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=172.16.2.0/24,action=move:NXM_OF_ETH_DST[]->NXM_OF_ETH_SRC[],resubmit(,1)"
OVS1 # ovs-ofctl add-flow br0 "table=1,ip,action=load:2560->NXM_OF_IP_DST[16..31],load:44048->NXM_OF_IP_SRC[16..31],resubmit(,2)"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.0.0/24,nw_src=172.16.0.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.1.0/24,nw_src=172.16.1.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.0.0/24,nw_src=172.16.1.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.1.0/24,nw_src=172.16.0.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.2.0/24,nw_src=172.16.2.0/24,action=mod_dl_dst:<MAC_ADDRESS_R4>,IN_PORT"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.0.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2 >,output:1"
OVS1# ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.1.0/24,action=mod_dl_dst:<MAC_ADDRESS_R2 >,output:1"
OVS1 # ovs-ofctl add-flow br0 "table=2,ip,nw_dst=10.0.2.0/24,action=mod_dl_dst:<MAC_ADDRESS_R4 >,output:2"

Figure 7.13: OVS1 configuration for one-to-one mapping model.

7.5.1.2 Cell block based Mapping

Consider a graph modelling the network, where nodes are hosts/devices and edges are links. To minimize the virtual IP addresses required for the proposed framework, we divide the whole network into cell blocks.

Definition 5 (**Cell Blocks**). *Given a network graph G, cell blocks of G is defined as set of connected components obtained after removal of SDN switches from G along with all the links incident on SDN switches.*

Consider a pair of hosts that wants to communicate, and both hosts belong to a different cell block. In this case, the path between given pair of hosts consists of at least one SDN switch which connects both the cell blocks. Thus, any pair of hosts which belong to different cell blocks does not require virtual IP address for waypoint enforcement and can communicate with each other with real IP addresses. However, hosts belonging to the same cell block have to use virtual IP addresses to achieve waypoint enforcement. Thus, we need virtual IP addresses only for the communications within the cell blocks and use real IP addresses to communicate with the hosts outside the cell block. Thus, we can reuse virtual IP addresses across cell blocks. In this model, the number of virtual IP addresses required is equal to the number of hosts in the largest connected component (i.e., a cell block). We have two cell blocks in the topology given in Figure 7.10. Cell block 1 contains subnet 1 and subnet 2 and cell block 2 contains subnet 3. Cell block 1 contains the largest number of hosts (hosts in subnet 1 and subnet 2). Let the virtual IP address space be 172.16.0.0/23. The mapping of real IP address to virtual IP address for each subnet is given in Table 7.3. As it is evident from Table 7.3, the virtual IP addresses in 172.16.0.0/24 are used in cell block 1 as well as in cell block 2.

Table 7.3: Cell lock based mapping

REAL IP	VIRTUAL IP
10.0.0.0/24	172.16.0.0/24
10.0.1.0/24	172.16.1.0/24
10.0.2.0/24	172.16.0.0/24

We need to configure the network to handle packets with virtual destination IP addresses and forward these packets to the closest SDN switch. Like in one-to-one mapping, R1 and R3 must forward packets to R2, and R2 and R4 must forward the packet to OVS1.

The configurations of R1, R2, R3, and R4 are similar to one-to-one mapping.

OVS1 Configuration

```
OVS1 # ovs-ofctl add-flow br0 table=0,priority=0,actions=NORMAL
OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=172.16.0.0/23,action=resubmit(,1)"
OVS1 # ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.0.0/24,action=load:2560->NXM_OF_IP_DST[16..31],
load:44048->NXM_OF_IP_SRC[16..31],resubmit(,2)"
OVS1 # ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.1.0/24,action=load:2560->NXM_OF_IP_DST[16..31],
load:44048->NXM_OF_IP_SRC[16..31],resubmit(,2)"
OVS1 # ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.2.0/24,action=load:655362->NXM_OF_IP_DST[8..31],
load:11276288->NXM_OF_IP_SRC[8..31],resubmit(,2)"
OVS1 # ovs-ofctl add-flow br0 "table=2,action=push:NXM_OF_DL_DST[],move:NXM_OF_DL_SRC[]->
NXM_OF_DL_DST[],pop:NXM_OF_DL_SRC[],IN_PORT"
```

Figure 7.14: OVS1 Configuration for cell block based mapping model.

The flow entries required at the OVS1 switch to handle packets with virtual IP addresses are given in Figure 7.14. OVS1 checks if the destination IP address belongs to the virtual IP address space. If not, the packet matches the first flow entry, which has an action to process the packet as a normal L2/L3 device. If yes, it sends the packet to table 1 for further processing. In table 1, based on the source IP address's subnet, it modifies the source address IP address (convert it to virtual IP address) and destination IP address (convert it to real IP address) and sends the packet to table 2. Table 2 swaps the source and destination MAC addresses and sends the packet back along the incoming port.

In both one-to-one mapping and cell block based mapping, we add routes to the routing table of legacy switches (Layer 3 switches) to forward the packets with virtual destination IP addresses to the closest SDN switch. Further, multiple subnets can be present in one cell block, which may increase the virtual addresses as well as routing entries. To solve this problem, we can do mapping based on subnets and use policy based routing to further reduce the virtual IP addresses required.

Table 7.4: Subnet based mapping

REAL IP	VIRTUAL IP
10.0.0.0/24	172.16.0.0/24
10.0.1.0/24	172.16.0.0/24
10.0.2.0/24	172.16.0.0/24

7.5.1.3 Subnet Based Mapping

Layer 3 switches in the network support policy based routing (route-maps, source based routing, and Access Control List (ACL)). We can leverage it to lower the virtual IP address requirement. Once the packet reaches the layer 3 switch, we can then use policy based routing (using ACLs and route-maps) to forward the packets towards the SDN switch. Once the packets reach the SDN switch and they are to be sent back towards the intended destination using IP based routing instead of policy based routing. When the destination host lies in the same subnet as the source host, the traffic packets directly reach the destination via the access switch using MAC-based forwarding and may not reach the layer 3 switch. Thus, we need to use virtual IP addresses for communications within the same subnet. Any packet with virtual IP address is sent to the subnet's default gateway, to which the communicating hosts belong. Thus, packets eventually reach the SDN switch. SDN switch converts the source IP address to virtual IP address, and virtual IP address of destination back to real and the packets are sent back to the intended destination using destination based IP routing. In the topology given in Figure 7.10, all subnets are of the same size. Let the virtual IP address space be 172.16.0.0/24. In this method, we reuse the virtual IP addresses across subnets as given in Table 7.4. The policies and configurations of R_1 and R_2 are given in Figure 7.15. Similarly, we configure R_3 and R_4. The configuration of OVS1 is given in Figure 7.16.

R1 Configurations	R2 Configurations
R1#configure terminal R1(config)#ip access-list extended 100 R1(config-ext-nacl)#permit ip any any R1(config-ext-nacl)#exit R1(config)#route-map PBR R1(config-route-map)#match ip address 100 R1(config-route-map)#set ip next-hop 10.0.3.2 R1(config-route-map)#exit R1(config)#interface f0/0 R1(config-if)#ip policy route-map PBR R1(config-if)#exit	R2#configure terminal R2(config)#ip access-list extended 100 R2(config-ext-nacl)#permit ip any any R2(config-ext-nacl)#exit R2(config)#route-map PBR R2(config-route-map)#match ip address 100 R2(config-route-map)#set ip next-hop 10.0.4.1 R2(config-route-map)#exit R2(config)#interface f0/1 R2(config-if)#ip policy route-map PBR R2(config-if)#exit R2(config)#interface f1/0 R2(config-if)#ip policy route-map PBR R2(config-if)#exit

Figure 7.15: Router R_1 and R_2 configuration for subnet based mapping model.

The virtual IP address is required only for the communications within the subnet.

R3 Configurations	R4 Configurations
R3#configure terminal R3(config)#ip access-list extended 100 R3(config-ext-nacl)#permit ip any any R3(config-ext-nacl)#exit R3(config)#route-map PBR R3(config-route-map)#match ip address 100 R3(config-route-map)#set ip next-hop 10.0.3.9 R3(config-route-map)#exit R3(config)#interface f0/0 R3(config-if)#ip policy route-map PBR	R4#configure terminal R4(config)#ip access-list extended 100 R4(config-ext-nacl)#permit ip any any R4(config-ext-nacl)#exit R4(config)#route-map PBR R4(config-route-map)#match ip address 100 R4(config-route-map)#set ip next-hop 10.0.4.1 R4(config-route-map)#exit R4(config)#interface f0/1 R4(config-if)#ip policy route-map PBR R4(config-if)#exit

OVS1 Configuration
OVS1 # ovs-ofctl add-flow br0 table=0,priority=0,actions=NORMAL OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.1.0/24,nw_src=10.0.0.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT" OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.0.0/24,nw_src=10.0.1.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R2>,IN_PORT" OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.1.0/24,nw_src=10.0.2.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R2>,output:1" OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.0.0/24,nw_src=10.0.2.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R2>,output:1" OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.2.0/24,nw_src=10.0.0.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R4>,output:2" OVS1 # ovs-ofctl add-flow br0 "table=0,ip,nw_dst=10.0.2.0/24,nw_src=10.0.1.0/24,action=mod_dl_src:<MAC_ADDRESS_OVS1>, mod_dl_dst:<MAC_ADDRESS_R4>,output:2" OVS1# ovs-ofctl add-flow br0 "table=0,ip,nw_dst=172.16.0.0/24,action=resubmit(,1)" OVS1# ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.0.0/24,action=load:655360->NXM_OF_IP_DST[8..31],load:11276288->NXM_OF_IP_SRC[8..31], resubmit(,2)" OVS1# ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.1.0/24,action=load:655361->NXM_OF_IP_DST[8..31],load:11276288->NXM_OF_IP_SRC[8..31], resubmit(,2)" OVS1# ovs-ofctl add-flow br0 "table=1,ip,nw_src=10.0.2.0/24,action=load:655362->NXM_OF_IP_DST[8..31],load:11276288->NXM_OF_IP_SRC[8..31], resubmit(,2)" OVS1 # ovs-ofctl add-flow br0 "table=2,action=push:NXM_OF_ETH_DST[],move:NXM_OF_ETH_SRC[]->NXM_OF_ETH_DST[], pop:NXM_OF_ETH_SRC[],IN_PORT"

Figure 7.16: OVS1 configuration for subnet based mapping model.

Therefore, the virtual IP addresses can be reused for communications in other subnets. SDN switch must identify the subnet to which the host belongs and must know the mapping from host real IP address to virtual IP address and vice versa based on the subnet it identified. The maximum number of virtual IP addresses required in this model is equal to the size of the largest subnet. This model lowers the number of virtual IP addresses required but at the cost of increased policy definitions and is not easy to comprehend and visualize. The difference between subnet based mapping and cell block based mapping is that, virtual IP addresses can be reused across the subnets in case of subnet based mapping instead of cell blocks.

7.5.1.4 Dynamic IP Mapping

We can use IP based forwarding (routing tables) and still lower the virtual IP addresses by defining a dynamic mapping between hosts real IP addresses and virtual IP addresses. Hosts are given virtual IP addresses that expire over time. Heuristic: Hosts only communicate with a fewer number of hosts in the network. Using this heuristic, in a given window of time, we can determine the maximum number of hosts to which a given host may communicate and define the requirement for the number of virtual IP addresses. When a host demands for more virtual IP addresses than either deny the request or give the real IP address of the host (this can result in no waypoint enforcement for that flow). Similar to cell block based mapping, mapping of destination IP address depends on source IP address, however virtual IP address can change over time. For example, host A wants to communicate with host B. Host A requests for the IP address of Host B. Based on virtual IP address availability for Host A, SDN controller allocates virtual IP address for host B with respect to host A. When host A sends a packet with the virtual IP address of host B as destination address, the packet reaches the SDN switch. Now the SDN controller allocates a virtual IP for host A with respect to host B and changes the source address to this virtual IP address and destination address to that of host B's real IP address and forwards the packet to host B. Now host A and host B can communicate with each other using these virtual IP addresses. In dynamic IP mapping, the virtual to real IP address association can change over time. And this mapping is defined per host per destination basis, thus the number of flow entries in SDN switch increases.

7.5.2 Analysis of the Proposed Method

In this section we analyse the proposed framework with respect to load on the link and increase in average path length of communications in the network.

7.5.2.1 Network Set-Up

To analyse the performance of the proposed solution, we consider a 3-tier hierarchical network topology suggested by Cisco guideline [135], as shown in Figure 7.17. In network topology, layer 2 switches are used at access layer and layer 3 switches are used at core

and distribution layers. The network has a large number of access layer devices and a few high-end core devices. This indicates that a few legacy switches of distribution layer can be replaced with SDN switches. We consider a symmetric topology for ease of analysis.

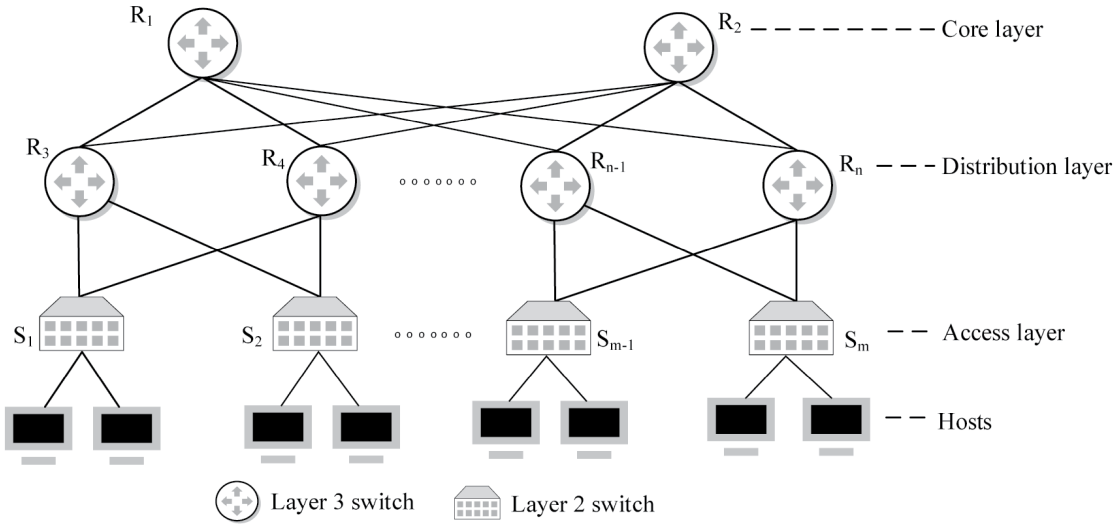

Figure 7.17: 3-tier Network Topology.

7.5.2.2 Assumptions

1. All links in the network are full duplex, and the capacity of all the links at a given layer is same.

2. All hosts generate the same unit amount of traffic. We divide the traffic generated by the hosts into three different categories:

   (a) Access Traffic (T_a): The percentage of total traffic generated by hosts, which is resolved by an access layer switch, is called *access traffic*.

   (b) Distribution Traffic (T_b): The percentage of total traffic generated by hosts, which is resolved by a distribution layer switch, is called *distribution traffic*.

   (c) Core Traffic (T_c): The percentage of total traffic generated by hosts, which is resolved by a core layer switch, is called *core traffic*.

3. In our analysis, we consider that traffic composition (T_a, T_d, T_c) for all hosts is same. Also, when a host generates packets of a given type, all possible destinations are equally likely. Based on the traffic composition, the likelihood that a packet travels

Table 7.5: List of symbols used in framework 1

Symbol	Meaning
S_c	Number of core switches
S_d	Number of distribution switch pairs
S_a	Number of access switches connected to one distribution switch pair
A_h	Number of hosts connected to one access switch
T_a	Percentage of total traffic resolved at access layer
T_d	Percentage of total traffic resolved at distribution layer
T_c	Percentage of total traffic resolved at core layer
HA_{load}	Load on the links in host-access layer
AD_{load}	Load on the links in access-distribution layer
DC_{Load}	Load on the links in distribution-core layer
N	Total number of distribution switches.
S	Total number of SDN switches at distribution layer
L	Total number of legacy switches at distribution layer
A_{plen}	Average path length of all traffic
AT_{plen}	Average path length of access traffic
DT_{plen}	Average path length of distribution traffic
CT_{plen}	Average path length of core traffic
DP_{plen}	Diverted path length
NP_{plen}	Native path length

through a given link varies from link to link. Considering no link acts as a bottleneck, for each host, we can find the likelihood that the packet flows through a given link. Table 7.5 lists the symbols used in the analysis.

7.5.2.3 Load on Link

Load on a link is defined as the total amount of traffic flowing through a given link in unit time when all host generates unit amount of traffic.

First, we calculate the load on the links in each layer of the network when there is no waypoint enforcement. Let HA_{laod}, AD_{load}, and DC_{load} denote the load on the links in host-access layer, access-distribution layer, and distribution-core layer respectively.

$$HA_{load} = T_a + T_d + T_c = 1.0 \tag{7.8}$$

$$AD_{load} = A_h \times \frac{T_d + T_c}{2} \tag{7.9}$$

$$DC_{load} = S_a \times A_h \times \frac{T_c}{4} \tag{7.10}$$

All traffic generated by a host must flow through the host-access link, and each host generates a unit amount of traffic. Thus the load on each host-access link is also unity. Traffic from all hosts reaches the access switches. The distribution traffic and the core traffic is forwarded towards the distribution layer switches. Since there are two links from each access switch to distribution layer switches, the traffic is equally divided among these two links. Once the traffic reaches the distribution switches, the core traffic flows towards the core layer switches, and there are two links from each distribution switch to the core layer switches, the load is equally divided among these two links.

Now consider the case when SDN switches are placed in the distribution layer, and waypoint enforcement mechanism is enabled. Now, the traffic is diverted from its native path to the SDN switches. The amount of traffic passing through the host-access layer links remains the same. All the traffic reaching the access switches, including the access traffic, must be forwarded towards the distribution layer as SDN switches exist at the distribution layer. There are two kinds of links in distribution-core layer- (1) the links which are connected to the SDN switch (SDN Distribution-Core (SDN_{dc}) link) and (2) the links which are not connected to SDN switch (Legacy Distribution-Core ($Legacy_{dc}$) link). Load on these links is denoted by S_DC_{load} and L_DC_{load} respectively.

$$HA_{load} = T_a + T_d + T_c = 1.0 \tag{7.11}$$

$$AD_{load} = A_h \times \frac{T_a + T_d + T_c}{2} = \frac{A_h}{2} \tag{7.12}$$

$$L\_DC_{load} = S_a \times A_h \times \frac{T_a + T_d + T_c}{4} = \frac{S_a \times A_h}{4} \tag{7.13}$$

$$S\_DC_{load} = \frac{S_a \times A_h}{4} \times \{(T_a + T_d + T_c \times \frac{L-1}{N-1}) \times \frac{L}{S} + T_c\} \tag{7.14}$$

We use *Monte Carlo* simulations to validate the expressions derived for loads on various links in both the cases, with and without waypoint enforcement. We construct the network topology using the given network parameters (S_c, S_a, S_d, A_h). To find the load on a link, we first generate a fixed large number of packets from each host. Then based on the given traffic composition (T_a, T_d, T_c), we determine the type of each generated packet. Keep track of the number of packets that go through each link in a given direction. Once all packets are transmitted, we find the load on each type of link. For simulation, we consider topology parameter as, $S_c = 2, S_a = 4, S_d = 4, and A_h = 10$. We choose traffic composition as, T_a = 0.5, T_d = 0.25 and T_c = 0.25. Tables 7.6, 7.7, 7.8, and 7.9 show the comparison of theoretical and simulation results of load on different types of links with(-out) waypoint enforcement.

Table 7.6: Load on links without waypoint enforcement

Link Loads	Theoretical	Simulation
HA_{load}	1.000	1.00000
AD_{load}	2.500000	2.5013125
DC_{load}	2.500000	2.489625

Table 7.7: With waypoint Enforcement

Link Loads	Theoretical	Simulation
HA_{load}	1.000	1.00000
AD_{load}	5.000	5.0

Table 7.8: SDN_DC_{load} with waypoints

#Waypoints	1	2	3	4	5	6	7	8
Theoretical	70.000	30.35714	17.38095	11.07143	7.42857	5.11905	3.57143	2.5
Simulation	69.96375	30.34675	17.40933	11.05956	7.43005	5.11895	3.56478	2.50625

Table 7.9: $LEGACY\_DC_{load}$ with waypoints

#Waypoints	1	2	3	4	5	6	7
Theoretical	10.000	10.000	10.000	10.000	10.000	10.000	10.000
Simulation	9.99482	9.99966	10.01470	9.99681	10.00058	9.99987	9.8795

7.5.2.4 Path length

Path length is defined as the average number of hops a packet traverses before reaching its destination. Total average path length is given by,

$$A_{plen} = AT_{plen} \times T_a + DT_{plen} \times T_d + AT_{plen} \times T_c \tag{7.15}$$

where, AT_{plen}, DT_{plen}, and CT_{plen} are the average path lengths of access traffic, distribution traffic, and core traffic, respectively. When waypoint enforcement is enabled, the average path length of each type of traffic is a function of the number of SDN switches in the network. Also, enabling waypoint enforcement increases the path length. For a traffic type X, the average path length is calculated as follows,

$$X_{plen} = DP_{plen} \times P(D_X) + NP_{plen} \times (1 - P(D_X)) \tag{7.16}$$

where, DP_{plen} is the diverted path length and NP_{plen} is the native path length. $P(D_X)$ is the probability that packet of a given traffic type X is diverted for waypoint enforcement. Equations (7.17), (7.18), (7.19) give the probability of divergence for each type of traffic. $P(D_a)$, $P(D_d)$, $P(D_c)$ denotes the probability of divergence of access layer traffic, distribution layer traffic, and core layer traffic respectively.

$$P(D_a) = 1 \tag{7.17}$$

$$P(D_d) = \frac{L}{L+S} \tag{7.18}$$

$$P(D_c) = \frac{L}{L+S} \times \frac{L-1}{L+S-1} \tag{7.19}$$

7.5.3 Results

In this section, we present the analytical results of the proposed solution. To perform the analysis, we consider a 3-tier architecture as suggested Cisco, with the following topology and traffic composition parameters: $S_c = 2, S_d = 20, S_a = 10, A_h = 20$ and $T_a = 0.5$,

$T_d = 0.25$, $T_c = 0.25$.

7.5.3.1 Load on Different Links

We replace the legacy switches with SDN switches at distribution layer. Figure 7.18 (a), shows that the SDN distribution-core links experience high load due to traffic divergence from legacy distribution-core links. The additional load on SDN distribution-core links can further be divided into two types- (1) the access traffic and the distribution traffic that reaches the distribution switch which is not an SDN switch and must be forwarded to an SDN switch, (2) the core traffic between two distribution switches when neither of the distribution switches is an SDN switch and must be forwarded to an SDN switch. We call the load exerted by these traffic on the SDN distribution-core links as *internal deviation load* and *core deviation load*, respectively. As shown in Figure 7.18 (a), these loads decrease with increase in number of SDN switches. And eventually reduce to zero when all distribution switches are replaced with SDN switches.

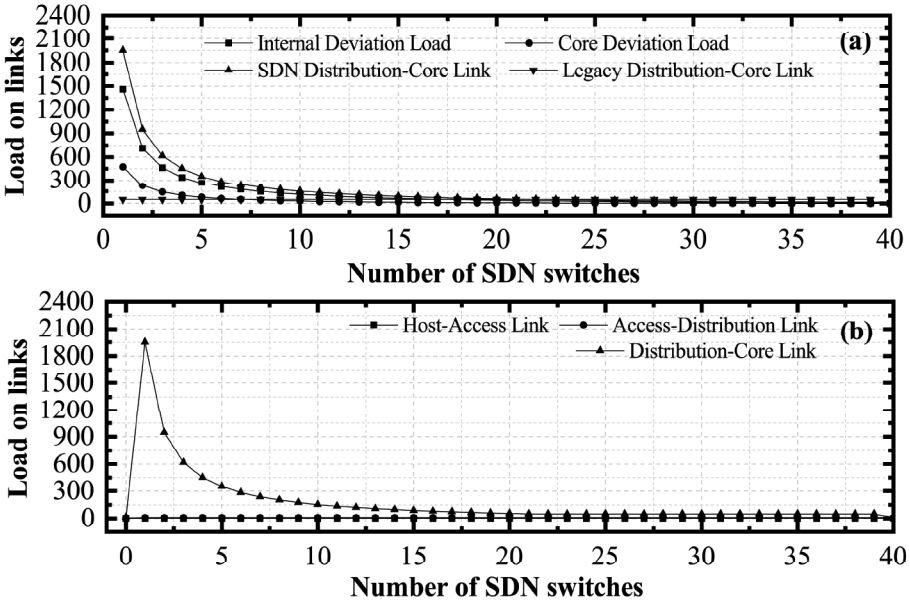

Figure 7.18: Load on the links at each layer

Figure 7.18 (b) shows the maximum load on the links in different layers of the network. The load on host-access link and access-distribution link remains the same because entire traffic of underlying hosts has to go to/through the distribution layer for waypoint enforcement. The load on distribution-core link decreases drastically as we increase

the number of SDN switches because access layer and distribution layer traffic is being resolved by the SDN switches at distribution layer.

7.5.3.2 Path length

Figure 7.19, shows the effect on average path length (in terms of number of hops) with respect to the number of SDN switches deployed in the network. The average path length is calculated using Equations 7.15 and 7.16. We observe that as we increase the number of SDN switches, the average path length decreases. Thereby reducing the propagation delay. Since we are replacing the distribution switches with SDN switches, all access traffic still must be forwarded towards distribution switches. So, even after replacing all distribution switches with SDN switches, the path length of waypoint enforced traffic does not become equal to the path length of the traffic in legacy network.

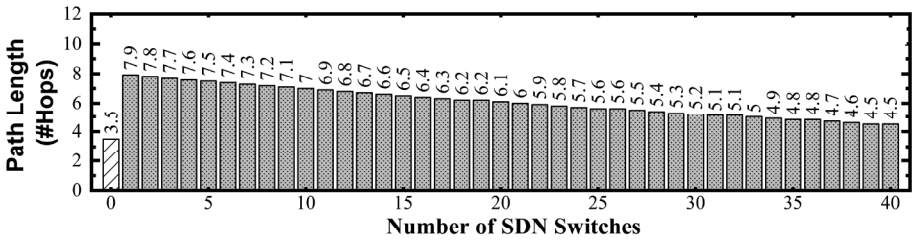

Figure 7.19: Average path length with respect to number of SDN switch.

7.6 Analysis and Performance Evaluation of All Methods

We evaluate all the solutions for average path length and percentage of waypoint enforcement achieved. Since distribution layer switches are being replaced incrementally, there are only a few SDN enabled switches in the network. Depending on the method [68, 66, 67, 224, 225] used to achieve waypoint enforcement, it may not be possible to divert the whole traffic through SDN switches. So, we measure the ratio of packets that traverse at least one SDN switch in their path to the total number of packets in the traffic to obtain the percentage of traffic begin enforced. That is,

$$P_{WPA} = (P_{SDN\_switches} \ / \ P_{total}) \times 100 \tag{7.20}$$

Where, P_{WPA} is the percentage of waypoint enforcement achieved, $P_{SDN\_switches}$ is the number of packets going through the SDN switches, and P_{total} is the total number of packets in the network.

For simulation, we consider topology parameter as $S_c = 3, S_d = 206, S_a = 6, A_h = 10$. For the experiments, we use LBNL(Lawrence Berkeley National Laboratory) traffic [229, 230]. We consider three scenario for traffic, (1) access traffic is dominating (i.e., $T_c = .0354$, $T_d = .0795$, $T_a = .8851$) (2) distribution traffic is dominating (i.e., $T_c = .0354$, $T_d = .8851$, $T_a = .0795$) (3) core traffic is dominating (i.e., $T_c = .8936$, $T_d = .0918$, $T_a = .0146$). Let k denote the number of waypoints enforced, i.e., the number of SDN switches in the network at the distribution layer. The order of deployment of SDN switches from k = 0 to k = 412 is assumed to be from left to right at distribution layer.

As shown in Figure 7.20, for all approaches, the percentage of waypoint enforcement increases with increase in k. Virtual IP based approach always provides 100% waypoint enforcement irrespective of k, as shown in Figure 7.20. For Telekinesis, the percentage of waypoint enforcement achieved in all three traffic scenarios never goes to 100% as it does not enforce the access layer traffic towards the SDN switch. When all the distribution layer switches are replaced with SDN switches Telekinesis achieves, 11.49% waypoint enforcement when access layer traffic is dominating as shown in Figure 7.20 (a), 92.05% when distribution layer traffic is dominating as shown in Figure 7.20 (b), 98.54% when core layer traffic is dominating as shown in Figure 7.20 (c). Magneto has a similar trend, but it saturates above Telekinesis [66] as it can enforce the access layer traffic to go through an SDN switch. Magneto does not achieve 100% waypoint enforcement as it cannot enforce the traffic of a subnet that does not have an SDN switch. Also, the waypoint enforcement depends on the availability of poisoned ARP entry in a host's ARP table. It is possible that ARP table entries corresponding to gratuitous ARP packets in the host machine are outdated and hence not used. Due to the same reason, Multi-VLAN [224] cannot achieve 100% waypoint enforcement. For Panopticon [68], we have 256 SDN_c ports per cell block. If there are less number of SDN switches in the network, there will be less number of cell blocks. Thus, the percentage of waypoint enforcement increases with the increase in the number of SDN switches, as shown in Figure 7.20.

The average path length is normalised w.r.t. its value when there is no SDN switch

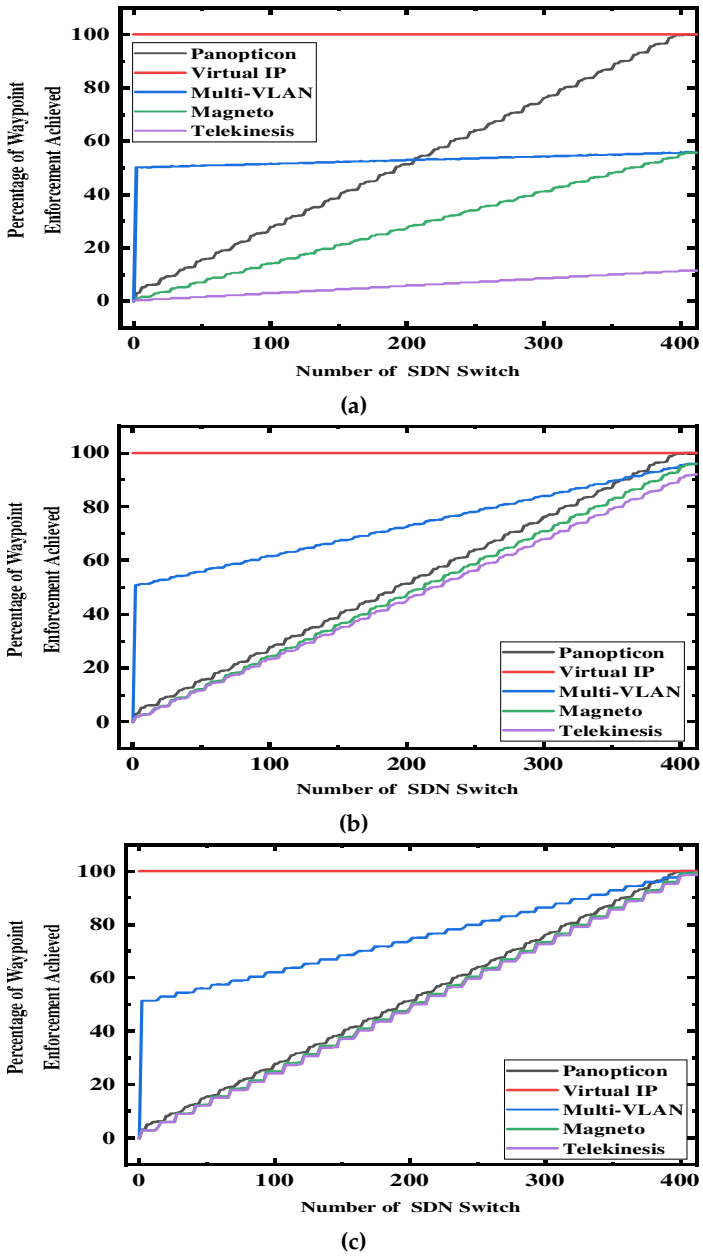

Figure 7.20: Percentage of Waypoint Enforcement Achieved. (a) When the access traffic is dominating (i.e., $T_c = .0354$, $T_d = .0795$, $T_a = .8851$) (b) When the distribution traffic is dominating (i.e., $T_c = .0354$, $T_d = .8851$, $T_a = .0795$) (c) When core traffic is dominating (i.e., $T_c = .8936$, $T_d = .0918$, $T_a = .0146$).

in the network. In all three traffic scenarios, discussed above, the average path length for Virtual IP based approach is highest as it always tries to achieve 100% waypoint enforcement irrespective of the value of k. The average path length decreases with increase in the number of SDN switches. In Multi-VLAN [224] (i.e., framework 2 proposed in Section 7.4), the traffic enforcement is done only if there is an ARP entry at the source host which maps the IP address of the destination host with MAC address of the core switch. In all

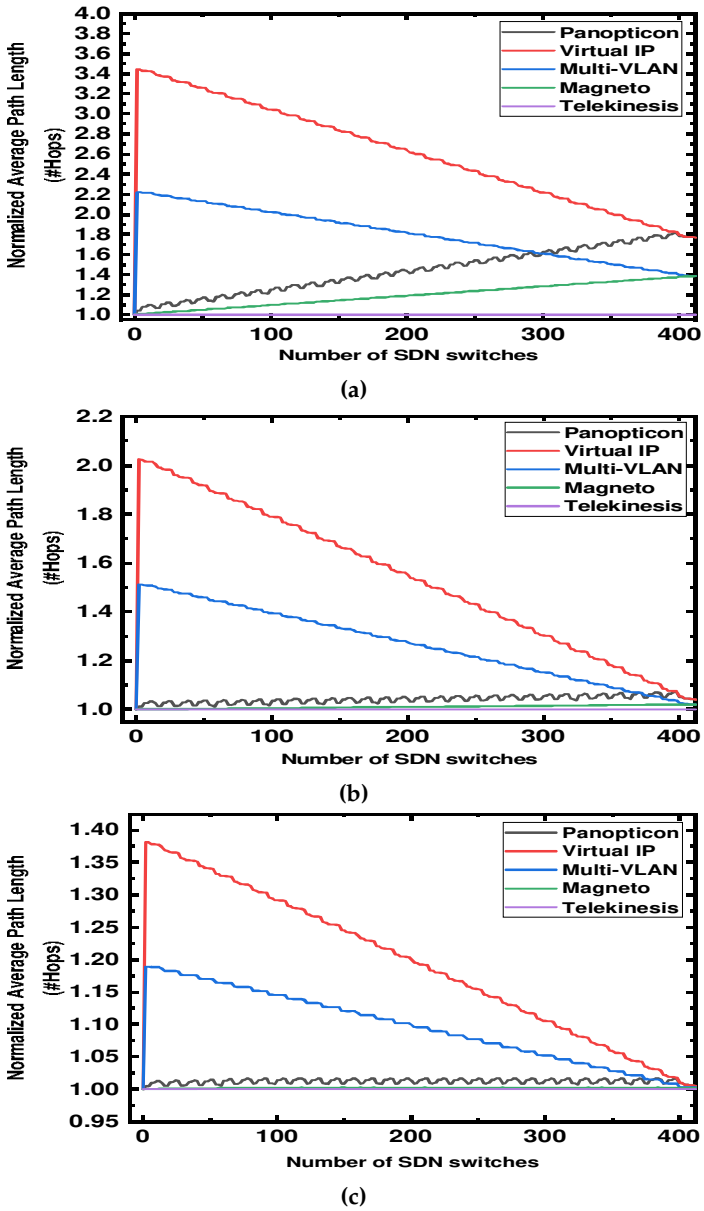

Figure 7.21: Normalized average path length. (a) When the access traffic is dominating (i.e., $T_c = .0354$, $T_d = .0795$, $T_a = .8851$) (b) When the distribution traffic is dominating (i.e., $T_c = .0354$, $T_d = .8851$, $T_a = .0795$) (c) When core traffic is dominating (i.e., $T_c = .8936$, $T_d = .0918$, $T_a = .0146$).

three traffic scenarios, we have taken three probabilities, 0.25, 0.50, and 0.75, for the ARP entry availability in the source host's ARP table. We have performed the experiments for all three probabilities for all three traffic scenarios mentioned above. The trend is similar with all three probabilities. That is, the average path length decreases with an increase in the number of SDN switches. We present the results only for 0.50 probability.

Panopticon [68], Telekinesis [66], and Magneto [67] does not guarantee 100% traffic

waypoint enforcement. The average path length increases with an increase in the number of SDN switches deployed. The ratio of increase in average path length is varying w.r.t. to the ratio of traffic. As shown in Figures 7.21 (a) and 7.21 (b), increase in average path length for the first scenario (when the access traffic is dominating) w.r.t. SDN switches deployed is more as compared to the second scenario (when the distribution traffic is dominating). The reason is that the access traffic which was previously being resolved at the access layer now has to go to the distribution layer or core layer. In the third traffic scenario (when the core traffic is dominating), most of the traffic is going through the distribution layer itself. Thus it gives less average path length, as shown in Figure 7.21 (c) compared to the first and second traffic scenarios, as shown in Figure 7.21 (a) and (b), respectively.

7.7 Summary

In Hybrid SDN networks, waypoint enforcement is essential to gain the benefits of SDN. Achieving full waypoint enforcement in a Hybrid SDN network becomes challenging as all the packets travelling in a network may not necessarily pass through any of the SDN switches present in the network. To the best of our knowledge, there is no solution in the literature which provides full waypoint enforcement. In this chapter, we proposed a novel framework to achieve full waypoint enforcement using virtual IP addresses. It provides full waypoint enforcement even with a single SDN switch in the network. Since it achieves full waypoint enforcement, the average path length is more compared to the existing solution. It also provides flexibility to the administrator to decide the percentage of waypoint enforcement she wants to achieve. Based on the requirements, she can define the IP address mapping in the controller such that it provides real-IP addresses to the traffic that does not require waypoint enforcement. We propose another framework that overcomes the limitation of Telekinesis [66] and Magneto [67].