CARLETON UNIVERSITY

SCHOOL OF COMPUTER SCIENCE

AND

UNIVERSITY OF OTTAWA

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Improving Mobile IP Protocol to Handle Mobility in SDN

COMP 5903 (CSI 6900) INT. Graduate Project COM.SC.

Supervised by Prof. Amiya Nayak

Submitted By:

SHARRULATHA KATHIRVEL (101020208)
CARLETON UNIVERSITY
sharrulathakathirvel@cmail.carleton.ca

April 25, 2017

Abstract

The current Mobile IP Protocol is one of the most used mobility management protocols. A mobile node MN has a home address HoA which is registered in a home agent that serves as a mobility anchor point. When the MN moves, it acquires a care-of-address CoA, which is notified to the home agent by means of on-going tunnel communication to the new location of MN. The above-mentioned protocol works well, but with the increase amount of mobility, it leads to issues such as triangle routing, higher latency, or session disruption.

Software Defined Networking promises to improve a network overall performance specific to mobility in terms of Handover latency, signaling overhead. In this project, we proposed a solution for IP mobility based on SDN to update the switches with a new flow entry. The simulations of the topology are implemented by Mininet emulator.

Keywords: SDN, OpenFlow, Mininet, POX controller, Host, OVSSwitch, TCP Sockets

Contents

1. Introduction

With the increase in popularity of mobile devices and the demand to provide network access to mobile users with maximum throughput and minimal delay is the main goal of network companies. To handle this mobility of devices, Mobile IP [1] is one of the earliest and most well-known protocol. Mobile IP [1] is an open standard defined by the Internet Engineering Task Force (IETF) RFC 2002, that allow users to keep the same IP address, stay connected and maintain ongoing applications while roaming between IP networks [6]. In normal IP networks, routing is based on stationary IP addresses. The issues start when a device roams away from its home network and is no longer reachable using normal IP routing and Mobile IP [1] protocol resolves the issues in reaching the mobile device.

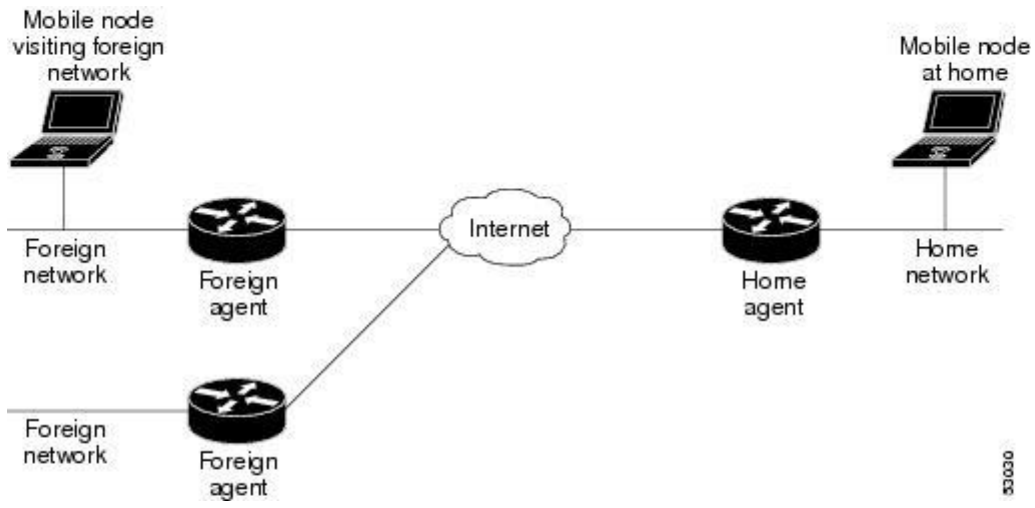


Figure 1: Mobile IP Components and Relationships [6]

Mobile IP [1] uses Home address (HoA) to identify a MN and Care-of-Address (CoA) to locate a MN. The components of the Mobile IP protocol are mentioned in Figure 1. Mobile IP [1] introduces a mobility anchor point called Home Agent (HA) to store the binding cache of each MN in the form of a mapping from the MN's HoA to its current CoA. With this binding cache, HA is responsible for tunneling packets to and from the MN. But the issues with the fixed HA is that it brings the triangle routing problem when the MN is away from HA. Triangle routing results in the data path stretch between two communicating devices because the entity that stores the MN's binding cache is no longer on the shortest path between MN and Correspondent nodes CN and all the packets MN to CN must take a detour to pass the HA. This triangle routing also leads to handover signaling cost as well as heavy load on HA due same reason.

In this report, we are addressing the above problem using SDN and Openflow. Software Defined Networking is an emerging network architectural approach while Openflow is one of the well-known instantiations of it [4]. In SDN architecture, the control and data planes are decoupled network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications [7]. In SDN, Network structures, functions and performance can be defined in a simpler way which is usually achieved by providing programmable devices and a centralized control logic.

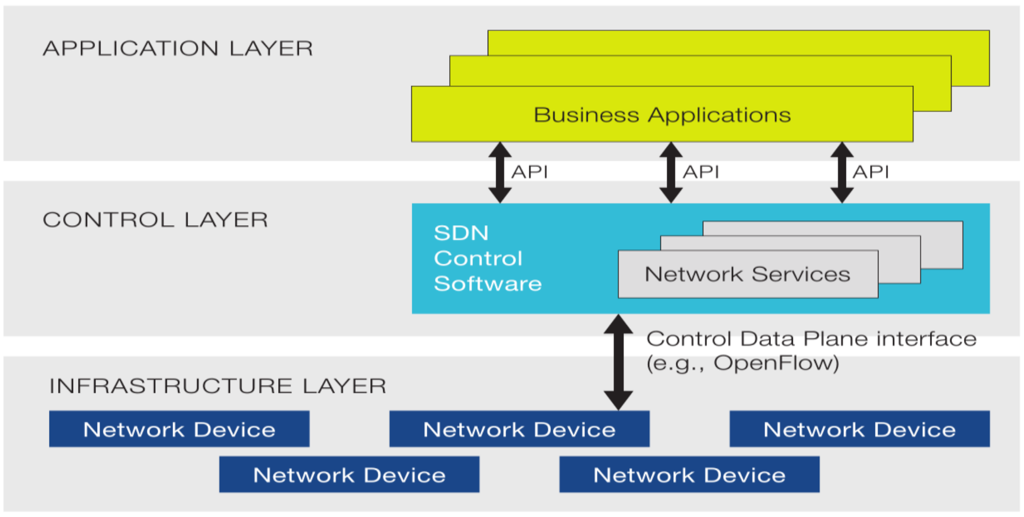


Figure 2: Software Defined Network Architecture [7]

A SDN architecture consists of several collaborating domains, each of which contains at least one controller. All the domains combined provides a large-scale mobility management service to its customers and there is a need for a mobility management application to manage the combined domains. It is deployed in collaborating controllers via SDN northbound interfaces. The controllers communicate with each other through an east/westbound interface to make the distributed mobility management application work and a southbound interface is employed to realize interactions between the control plane and data plane and it allows controllers to learn about the activities of host and to instruct SDN devices to forward packets to the hosts. Control plane has two functions: one is to maintain up-to-date identifier-to-locator mapping of each host and the other is to download a host mapping to SDN devices when necessary. The data plane must simply forward the incoming packet to the corresponding nodes but if it doesn't have information about the host it must request controller for the mapping. Figure 3 given below shows the mobility management architecture of SDN.

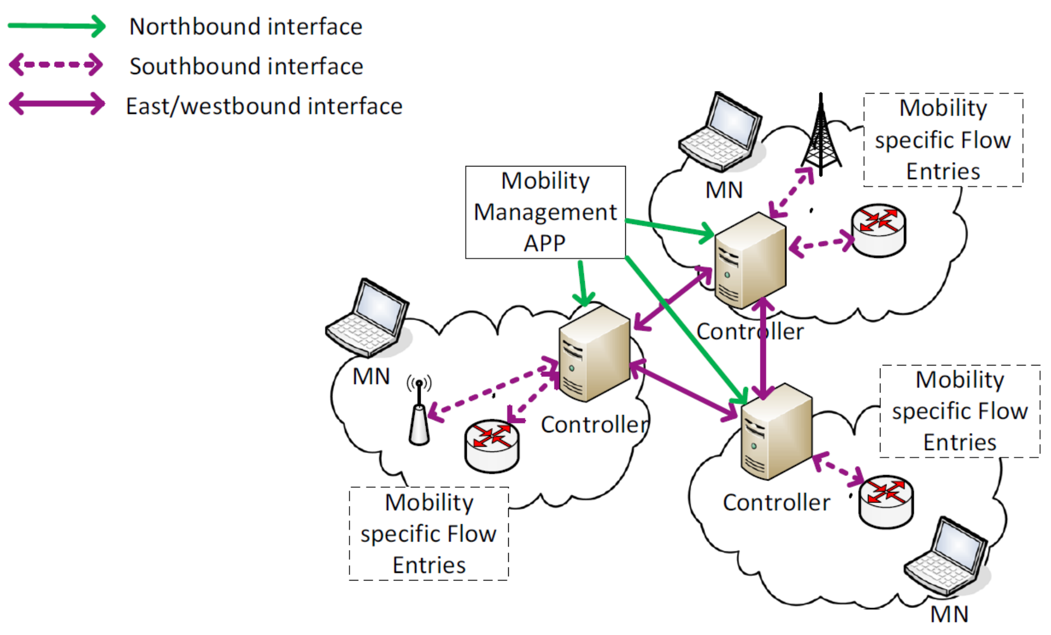


Figure 3: SDN Multi-Domain Mobility Architecture [3]

In the multi-domain SDN architecture, it clearly shows that all the control devices are inter-connected which helps the devices to find out the hosts when they move. SDN help to solve problems in IP mobility protocols, because firstly programmable network devices in SDN can alleviate or even eliminate triangle routing. It is because of the binding cache of MN and CN can be flexibly placed on the shortest path between the MN and CN instead of some fixed HA. Second, centralized controller architecture of SDN helps to reduce the complexity of the protocol.

2. Related Work

Many efforts have been paid to apply SDN to mobile and wireless networks. Recently, along with the development of 3GPP Long Term Evolution (LTE), there is a going trend to provide a more flexible and dynamic management, which is also a concern in the Internet research area. Mobile IP [1] centralizes both mobility signaling and data forwarding functions into a single HA, which increases signaling cost and data path stretch when MN is not within the home network. Researchers have begun studying on how to offer better mobility support under SDN architecture. To improve robustness during mobility handover, Yap et al [8] [9] proposed OpenRoads using multicast in Openflow networks. Pupatwibul [10] proposed a solution to enhance mobile IP networks using Openflow.

You et al [2] proposed a solution for IP Mobility support in Software Defined Networks. In [3], You et al implemented the proposed solution in [2] and evaluated the proposed solution in comparison with other algorithms. The focus of [3] is to integrate SDN and the current internet architecture. As like the mobility protocols, in [2] [3] the author assign a stable identifier HoA to each MN. In [2] [3], HoA is different from other proposed solutions, it is non-routable and belongs to a specific address block. The authors kept it in such a way so that we can use a MN's HoA to lookup its up-to-date location, other than to reach its home network which leads to potential triangle routing. In the proposed solution of [2] [3] the authors kept the first-hop open flow switch of a mobile node as CoA which also provides an advantage that when a MN moves to new network it never require to re-configure its IP address and the CoA address is routable.

In [2] [3], Openflow controller is responsible for maintaining binding cache which maps a MN's HoA to CoA. For each MN, a subset of Openflow switches in the network serve as indirection point for the MN. They store replica of the MN's binding cache in the form of flow table which is downloaded from the controller, and redirect packets toward the MN per the flow table. Figure 3 given below explains the topology diagram of [2].

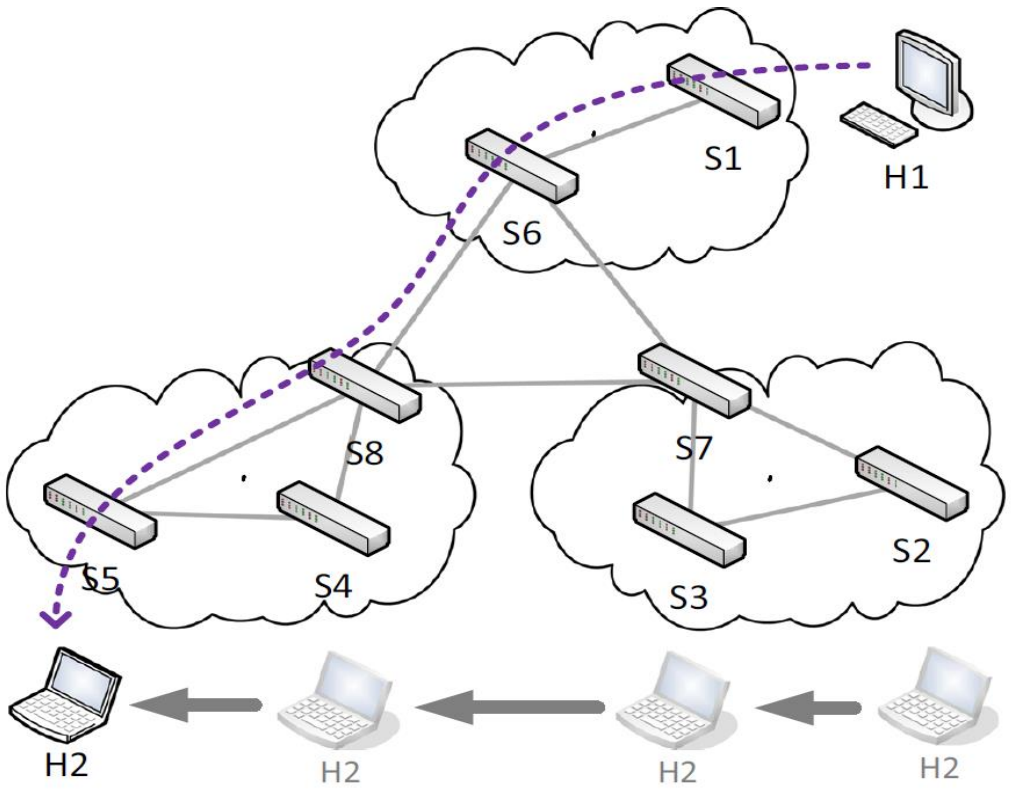


Figure 4: Multi-Domain SDN topology evaluated in [3]

In [2], the authors compared their proposed protocol with other protocols. In the above topology, the authors used POX [11] for Northbound interface, Openflow [4] as southbound interface, TCP sockets as East-West bound interface. In the idea proposed, the authors used IP address of MN as its identifier and used the address prefix of the MN's first hop Openflow switch as its locator. Therefore, once a MN roams to a new subnet, it keeps its own IP address unchanged and is assigned another routable address with the same prefix in the subnet. In [2] [3], the authors explained the mobility scenario in single controller and multiple controller cases. Figure 5 below explains the protocol flow proposed by authors in [2] [3]. The authors used Mininet [12] for creating the desired environment. In this project, we will carry out the simulations using Mininet and the controller that we used as control plane is POX [11] to find out the delay period when a host moves from one doma in to another domain.

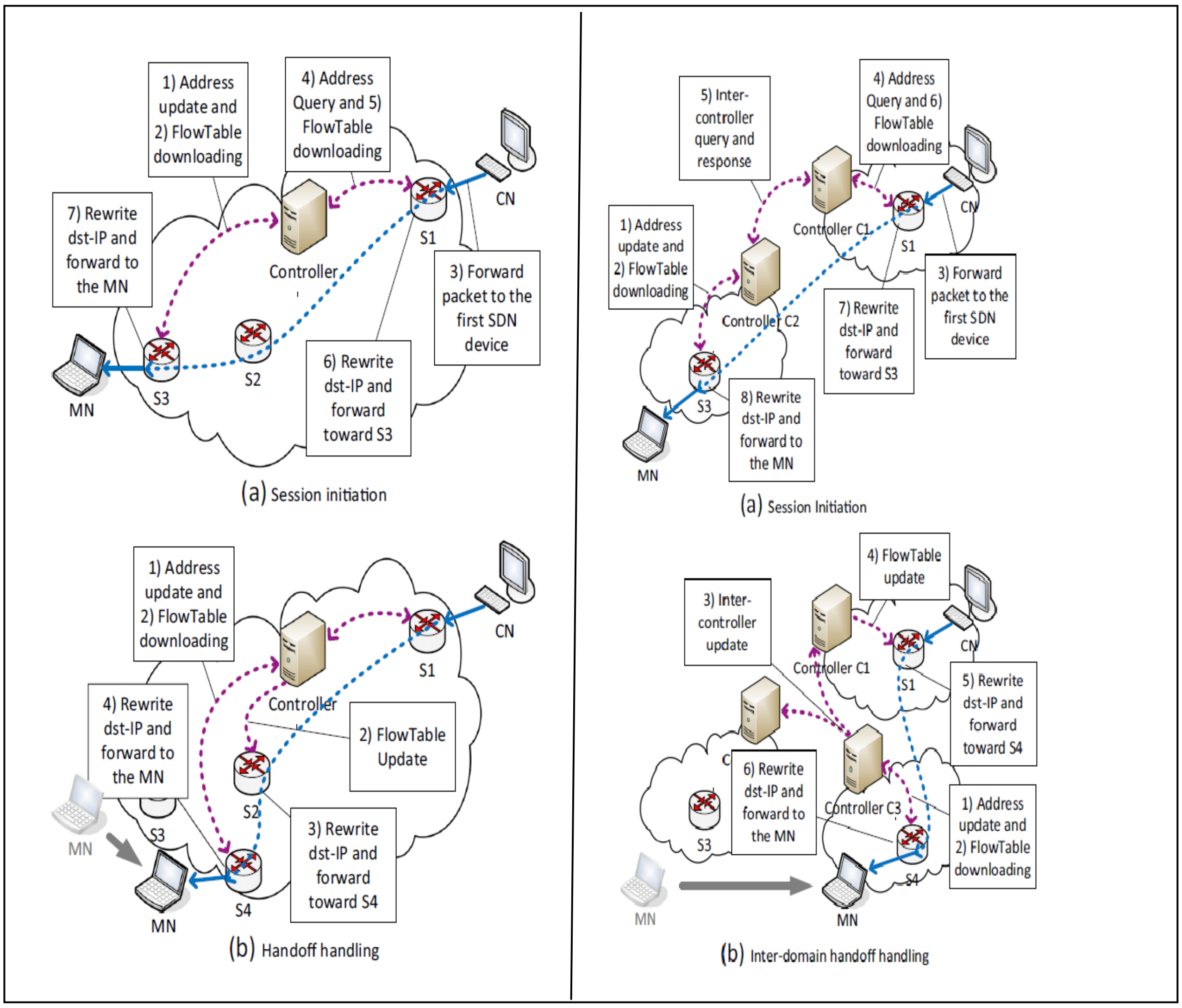

1) Address update and 2) FlowTable downloading
4) Address Query and 5) FlowTable downloading
7) Rewrite dst-IP and forward to the MN
Controller
CN
S1
S2
S3
MN
3) Forward packet to the first SDN device
6) Rewrite dst-IP and forward toward S3
(a) Session initiation
1) Address update and 2) FlowTable downloading
4) Rewrite dst-IP and forward to the MN
Controller
2) FlowTable Update
3) Rewrite dst-IP and forward toward S4
CN
S1
S2
S3
S4
MN
MN
(b) Handoff handling
5) Inter-controller query and response
4) Address Query and 6) FlowTable downloading
1) Address update and 2) FlowTable downloading
Controller C1
Controller C2
CN
S1
S3
MN
3) Forward packet to the first SDN device
7) Rewrite dst-IP and forward toward S3
8) Rewrite dst-IP and forward to the MN
(a) Session Initiation
3) Inter-controller update
4) FlowTable update
Controller C1
Controller C3
CN
S1
S3
S4
5) Rewrite dst-IP and forward toward S4
6) Rewrite dst-IP and forward to the MN
1) Address update and 2) FlowTable downloading
MN
MN
(b) Inter-domain handoff handling

Figure 5: (i) Single Controller (ii) Multi-Controller Domain Mobility handover in [2] [3]7

3. Problem Statement

3.1 Problem Definition

For the project, we will do an analysis of IP mobility in Hierarchical SDN architecture by monitoring certain metrics to find the suitable solution for cellular networks. By starting analyzing vertical SDN architecture, we show the advantages and defects of this architecture and present a vertical controller architecture for potentially solve these current defects.

In IP mobility, handover latency and Packet Loss is a main crucial metric in a big topology. Handover latency is defined as time lapse between the handover procedure starts and when it ends. The handover latency measures the time elapses from the source controller sent handover information to the root controller when the host leaves and the destination controller sends the handover host attachment to the root controller. And Packet Loss per user describes the amount of Packet loss with handover for a user under certain environment. Comparing these metrics, we will find strength and weakness of vertical approach in SDN IP mobility.

Table 1: Measuring Metrics

Metrics	
1	Handover latency
2	Packet Loss

3.2 Analysis of Current Solutions

We considered an existing solution [2] [3] in which has a distributed controller. The topology of the existing solution is as shown in the above Figure 4. The existing topology's control plane consists of a POX controller and the Data plane is a Openflow switch and the hosts are connected to the data plane. In [3], the authors implemented the proposal based on Mininet 2.1.0 [12] and the mobility of the hosts is simulated by detaching the MN from one switch and attaching it t anther, which will trigger port-status messages to the controller. The hosts move one domain to another domain in the multi domain cases and it simply disconnects and connects to switches in the same network in single domain cases. In multi controller mobility, if the both controllers present in the same administrative domain then the problem is simpler because intra-domain communication between controllers are common. When the two controllers belong to different administrative domain then it requires a more complex inter-domain controller communication.

To optimize the cost of inter-domain communication, in [2][3] the authors proposed a Binding Cache placement solution. The goal of the solution is to keep optimal forwarding path, minimize the distance between MN and TS and minimize flow entry downloading per movement. The solution is to replace the flow entry in the Fork Node after which the route to MN splits to go to new and old locations. For the implementation of the algorithm the authors used the below mentioned networking components in distributed architecture.

POX Controller: Provides a framework for communicating with Openflow switches using Openflow protocol and it keeps information of all OF-Switches and Hosts.

Openflow-Switch: It forwards packets, routes packets, and updates the corresponding flow table entries.

Host: A node which is capable to communicate with other hosts in the network and able to move from one region to another region.

As for host mobility between two controllers, the existing procedure is shown in Figure 6 [2] [3]. When the host detach from a switch, the switch will trigger OF_PORT_STATUS message to controller. When the host attaches itself to another switch which belongs to a controller in another domain, the switch sends an OF_PORT_STATUS notification to the controller and the controller updates a mapping entry and it triggers an inter-controller communication and informs the other controllers about the current location of the host.

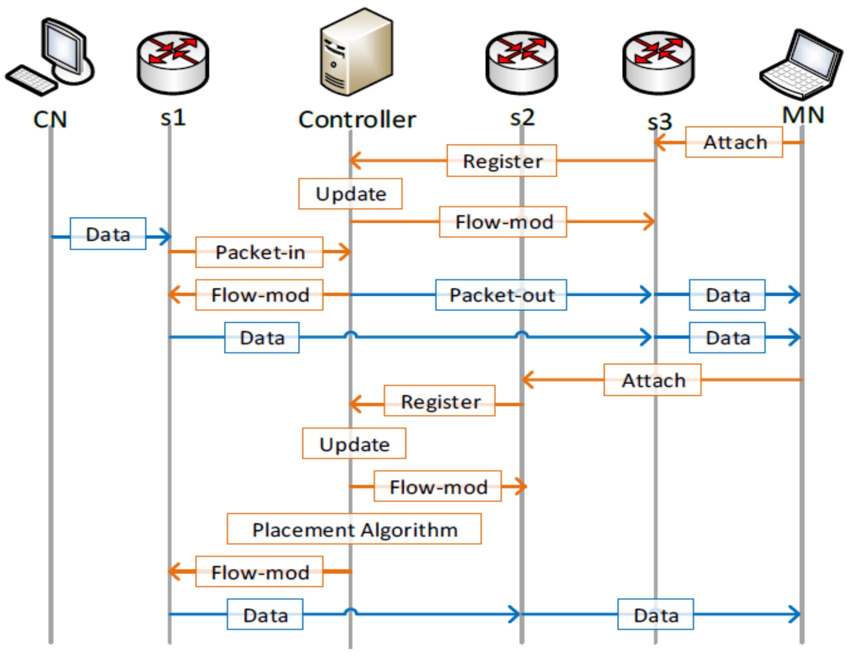


Figure 6: Protocol flow - Communication initiation and handover procedure [2] [3]

4. Proposed Solution

For this section, we will show the idea about the proposed solution, construct the architecture under certain assumptions, add key mechanism and analyze mobility solution.

4.1 Main Idea

We will analyze handover latency and packet loss of centralized SDN controller architecture. We are using a logically centralized root controller for the topology. With proper routing information from the controller, the hosts can communicate with the other hosts in the network. The main idea of the proposal is to avoid triangle routing, packet loss because of host mobility and to reduce the cost of inter-domain controller communication by maintaining all the details in a root controller.

4.2 Methodology

As shown in Figure 7, the centralized SDN topology consists of a Root controller, Sub-controllers, OF-Switch and hosts as its core parts. Root controller has the global view of the network, which includes controllers to switch Data Path ID (DPID) information, host IP address details and Switch to Host port pairing information. The main purpose of this topology is to reduce the cost of inter-domain controller communication. In [2] [3] the authors used a distributed SDN topology which has controllers, OF-Switch and hosts as its core parts. The disadvantage of the distributed topology is that when a host attaches to a switch, the controller must inform all other controllers about the status information. Since the root controller has the global view and all the information stored in the case of centralized architecture, the controller must inform only the Root controller and if other controllers need the host information, it can ask the root controller. By means of this the cost for inter-domain controller communication is cut into half.

In the proposed methodology, we use POX controllers for Root and Sub-controllers, Openflow switches and hosts as MN and CN. We use IP address of a MN as its identifier and use the address prefix of the MN's first-hop Openflow switch as its locator. Therefore, once a MN roams to a new subnet, it keeps its own IP address unchanged and is assigned another routable address with the same prefix in the subnet. Note that the MN is not aware of the routable address, but its first-hop switch handles the conversion between the routable address and the MN's identifier. We assume that only the identifier of a MN is known to its CNs. To ensure reachability of a MN, the first-hop switch of each CN needs to convert the MN's identifier to the routable address which is realized by a specific flow entry downloaded from the controller. Therefore, the controller is responsible for maintaining mappings between each MN's identifier and its current routable address. In the topology given below, there are certain switches called as anchor points. The switches that communicate with switches in another domain are called anchor point switches. In the topology given below the anchor points are S4, S5, S8, S9, S12, S13.

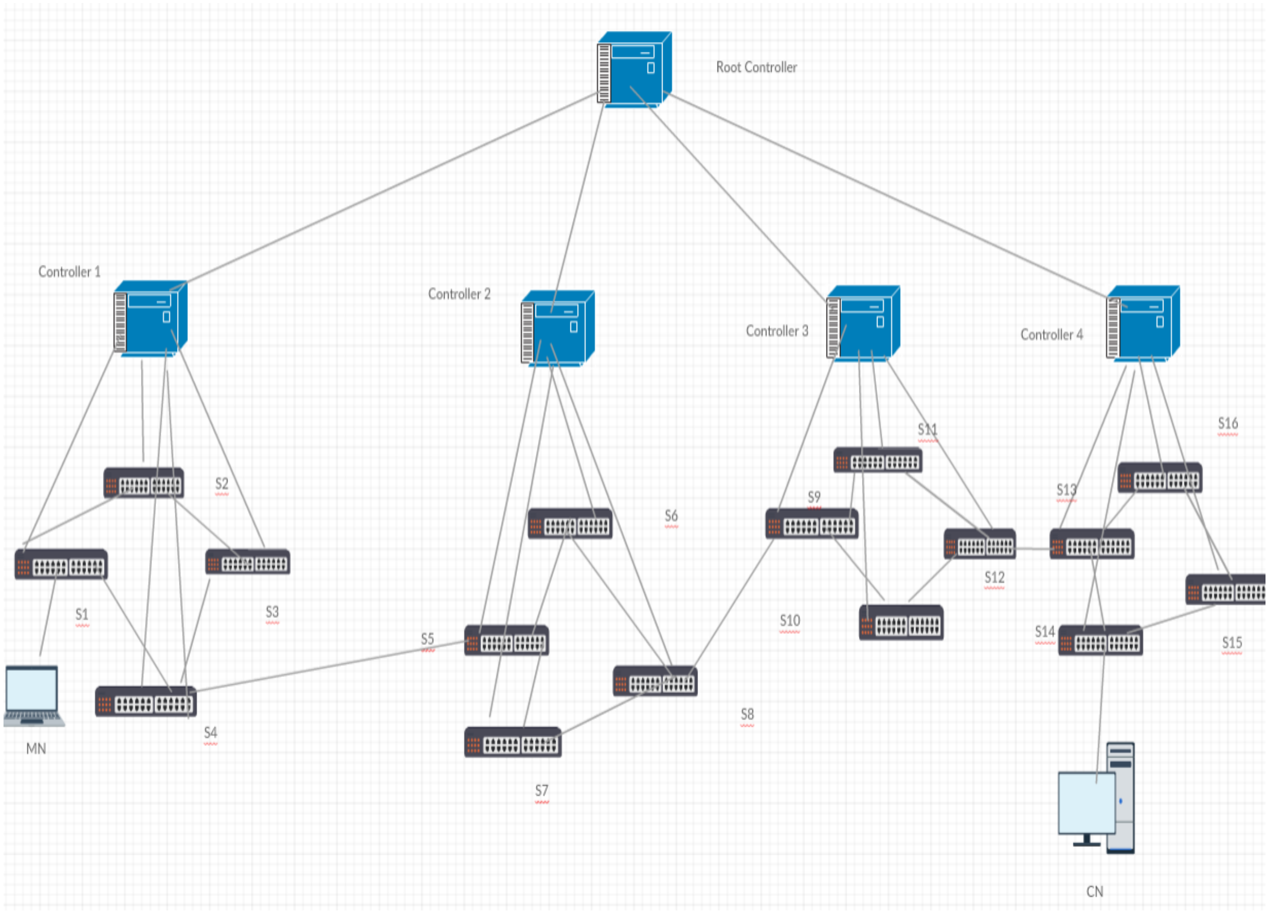


Figure 7: Centralized SDN Controller Topology

As shown in Figure 8, the centralized SDN topology consists of a Root controller (Cr), Sub-controllers (Ci, Cj, Ck) OF-Switch (Hsi, Fsi, Csk) and hosts (MNi, CNk) as its core parts. Initially there is a data communication between two hosts MNi and CNk. The root controller keeps track of the communication between hosts as well. So, it keeps track of communication between hosts MNi and CNk. After sometime, host disconnects from its home switch (Hsi) which triggers a status update message to its controller (Ci). After receiving the notification, the controller (Ci) triggers a communication to the controller which has the status update information. After receiving the information, the root controller updates its global table of hosts. At the same time the Mobile Node (MNi) attaches itself to a foreign switch (Fsi) which in turn triggers a port status message to its controller Cj. The controller will establish a communication to the root controller and inform about the new attachment. The root controller upon receiving the information will update its global table.

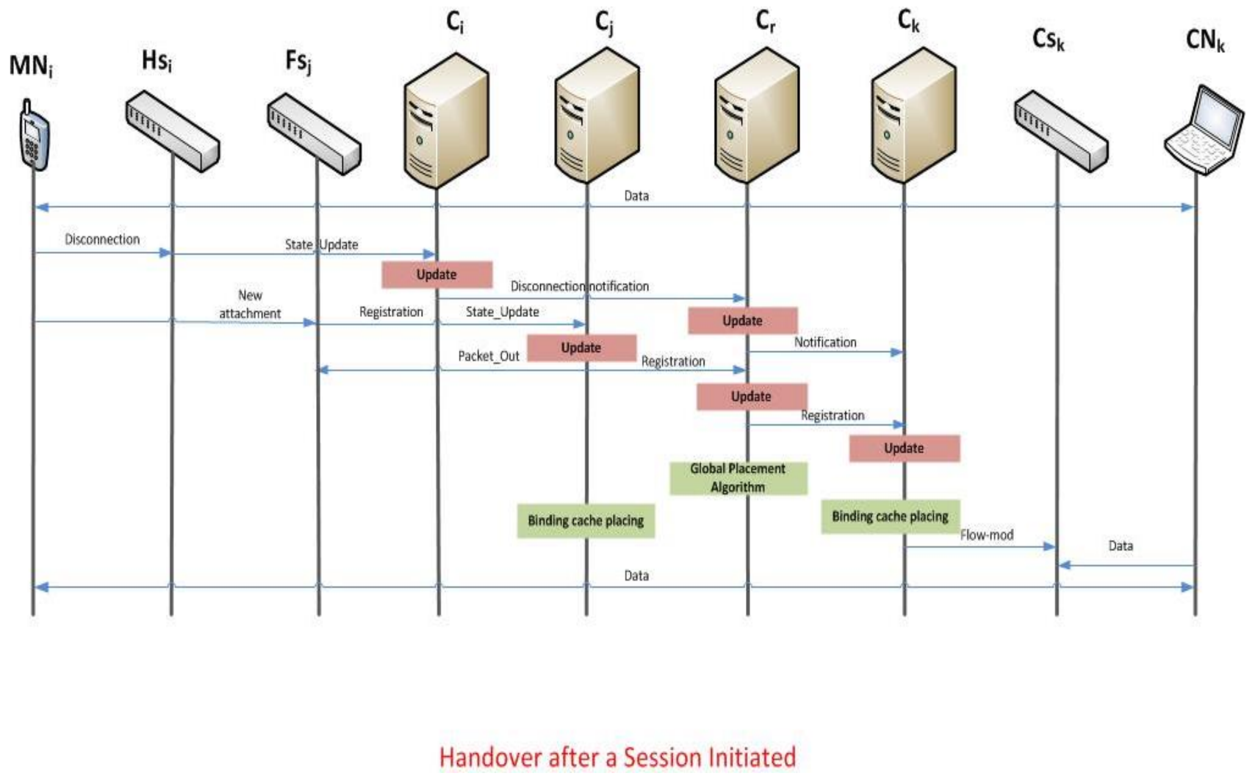


Figure 8: Handover Procedure in Centralized SDN topology

4.3 Binding Cache Placement

We are proposing a solution based on Anchor points. The root controller has all the information regarding the MN and CN communication. The root controller knows in which route the packets are travelling between MN and CN. Based on the route information, the Root controller selects the sub-controller in which it should replace the cache entry. The sub-controller then informs its anchor point to update the flow table. With proper routing information from the Root controller, placing it in a correct sub-controller cache placement and by updating the anchor points we can eliminate the triangle routing. By means of using the root controller reduces the communication cost between other controllers. Thereby we can achieve our goals, avoid triangle routing, packet loss because of host mobility and to reduce the cost of inter-domain controller communication by maintaining all the details in a root controller.

5. Simulation

5.1 Motivation

Our main aim is to prove that by using a Centralized SDN controller topology and the proposed binding cache algorithm we can avoid triangle routing, packet loss and cost reduction in inter-controller communication. By analyzing handover latency and packet loss, we want to show the performance of centralized controller architecture.

5.2 Experimental Setup

Mininet is a widely-used network emulator for researches in SDN. Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds [12]. Mininet 2.2.1 is the latest version. In this new version, it provides wired and wireless modules to support network simulation. With Mininet, we can easily interact with the network using Mininet CLI and API, create a customized topology and it provides a prototype for a SDN. In our experiment, we use the provided network modules and OpenFlow to simulate the centralized SDN architecture. In Mininet, we incorporated TCP socket stream code to establish the communication between controllers.

1) Download and Install required packages [12]

1. Download the Mininet VM image.
2. Download and install a virtualization system (VirtualBox).
3. Add the Mininet VM image in the VirtualBox
4. In the network preferences provide host only adapter configuration for the adapter of the image.
5. Boot up the VM machine
6. Login name: mininet Password: mininet
7. POX controller will be present as a built-in part of Mininet VM image

1. To make sure the interfaces exist use, ifconfig -a
2. Once you noticed the interfaces, assign IP address using DHCP, sudo dhclient eth1
3. The IP address assigned here will be used as the IP address of the controllers.

5.3 Simulation Environment

The centralized SDN topology is implemented in Mininet simulator. The topology diagram is mentioned in Figure 7. Here we utilize POX controller and Openflow which is already a part of Mininet. There are several API's in Mininet which helps us create a customized network topology. Below are some of the API which we used in simulating the customized topology. Before using them import the package mininet.topo which is available as a part of Mininet library.

1. For creation of hosts,

```
H1 = self.addHost('h1')
```

2. For creation of Openflow switches,

```
S1 = self.addSwitch('s1')
```

3. For creation of controllers, if the controller used is a remote controller, for example in our case it is POX, import RemoteController from Mininet library

```
Net = Mininet( Controller = RemoteController, Switch=OVSSwitch)
```

```
C1 = net.addController('c1')
```

4. After creating switches, hosts and controllers we must establish a link between them:

   To establish link between switch and host,

```
C1 = net.addLink(S1, H1)
```

   To associate a switch to a controller,

```
S1.start( [c1] )
```

5. After creating links between switches, hosts and controllers we have to build the network and start it.

```
Net.build() net.start()
```

6. If you want to start the CLI interface after creating the topology import the CLI package from Mininet library and add in your code,

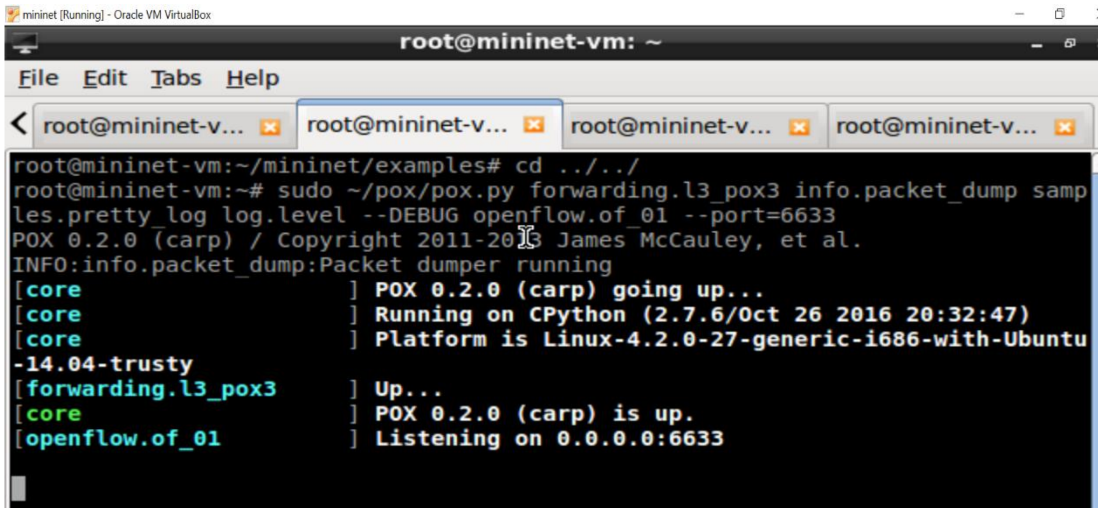
```
CLI( net )
```

7. While executing the code, you must start executing the controllers first, if you need multiple controllers first assign different IP and different Openflow ports,

```
C0 = RemoteController('c0', ip = '192.168.56.101', port =6633)
```

To execute the controller, match the Openflow port number while executing the POX controller. Figure 9 shows the output of the controller when it is executed,

```
sudo ~/pox/pox.py forwarding.l3_learning info.packet_dump samples.pretty_log
log.level --DEBUG misc.of_tutorial openflow.of_01--port=6633
```



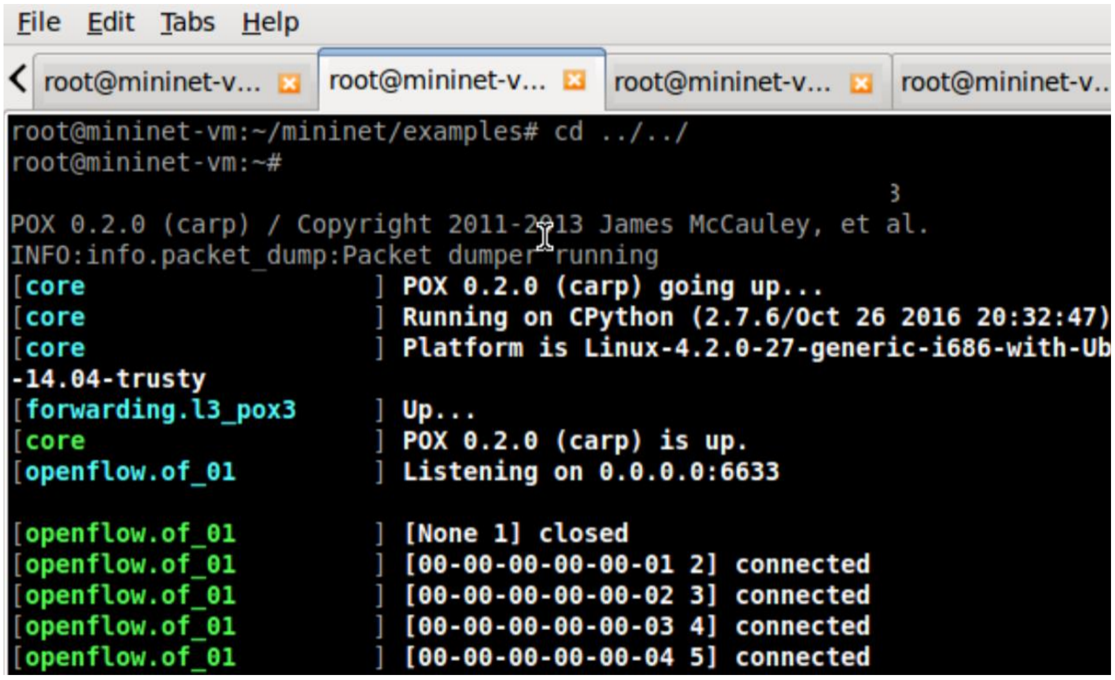
```
root@mininet-vm:~/mininet/examples# cd ../../
root@mininet-vm:~# sudo ~/pox/pox.py forwarding.l3_pox3 info.packet_dump samp
les.pretty_log log.level --DEBUG openflow.of_01 --port=6633
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:info.packet_dump:Packet dumper running
[core                   ] POX 0.2.0 (carp) going up...
[core                   ] Running on CPython (2.7.6/Oct 26 2016 20:32:47)
[core                   ] Platform is Linux-4.2.0-27-generic-i686-with-Ubuntu
-14.04-trusty
[forwarding.l3_pox3     ] Up...
[core                   ] POX 0.2.0 (carp) is up.
[openflow.of_01         ] Listening on 0.0.0.0:6633
```

Figure 9: POX controller execution output

8. After execution of POX controller, execute the customized topology script

```
sudo python topo.py
```

9. After execution of POX and topology script you will get to see a log info in the POX screen displaying DPID of the switch with a connected string. Below figure shows the connection establishment between controllers and switches.



```
root@mininet-vm:~/mininet/examples# cd ../../
root@mininet-vm:~#
                                                      3
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:info.packet_dump:Packet dumper running
[core                   ] POX 0.2.0 (carp) going up...
[core                   ] Running on CPython (2.7.6/Oct 26 2016 20:32:47)
[core                   ] Platform is Linux-4.2.0-27-generic-i686-with-Ub
-14.04-trusty
[forwarding.l3_pox3     ] Up...
[core                   ] POX 0.2.0 (carp) is up.
[openflow.of_01         ] Listening on 0.0.0.0:6633

[openflow.of_01         ] [None 1] closed
[openflow.of_01         ] [00-00-00-00-00-01 2] connected
[openflow.of_01         ] [00-00-00-00-00-02 3] connected
[openflow.of_01         ] [00-00-00-00-00-03 4] connected
[openflow.of_01         ] [00-00-00-00-00-04 5] connected
```

Figure 10: Connection establishment message between controller and switches

We designed the simulation environment in such a way that the host moves from one controller domain to another domain which is the mobility part. In Figure 7 is the initial topology setup which has 1 Root Controller, 4 Sub-Controllers. Each sub-controller has 4 switches each and each switch has a host. Initially MN is connected to switch S1, and it started communicating to a host CN which is under controller 4. In the simulation environment, we started iperf communication between MN and CN where CN as iperf server and MN as client. After a period, host MN moves to another switch S7 which is under controller 2. Now inter-domain mobility has happened.

First step, when the host detaches from S1, S1 will trigger the OF_PORT_STATUS message to controller1 after that controller1 will establish a TCP connection to the Root controller. Figure 10 shows the topology after MN moves to another controller. MN joins to switch S7 which will trigger an OF_PORT_STATUS message from S7 to controller 2. Now, controller2 will establish a TCP communication to root controller and root controller will do changes in the table (assumption). Now the root controller informs controller 2 to change the flow table of its anchor points to redirect the traffic to S7. After FLOW_MOD the packets are routed to S7 instead of S1 which avoids triangle routing.

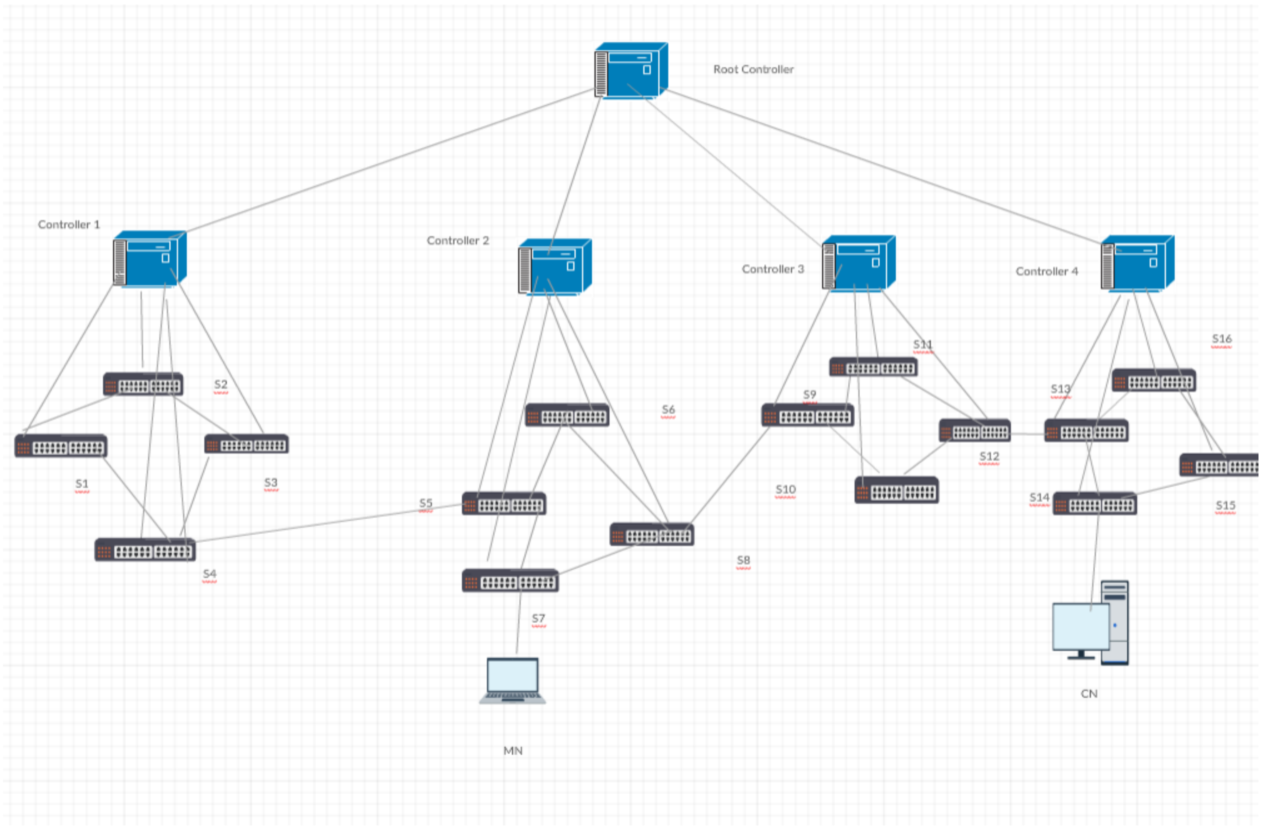


Figure 11: Topology after MN moves to another domain

5.4 Metrics

5.4.1 Handover latency

The handover latency is defined as the period used to perform handover procedure. It is a critical metric for measuring the handover procedure on what level it affects data transferring. Based on the topology in Figure 7 and Figure 10, it calculates the time cost for host MN to detach from switch S1 and to attach to switch S7. So, the handover latency (HL) can be calculated as follows:

For centralized SDN topology:

$$T_{HL}= T_{Port\_status\_attach\_Root:Handover\_Notify} - T_{Port\_status\_detach:Root\_detachment\_notify}$$

5.4.2 Packet Loss

When one or more packets of data travelling across a computer network fail to reach their destination, packet Loss happens. When a MN moves from one switch to another switch the ongoing communication between two hosts MN and CN results in some packet loss. During this packet loss period, MN is trying to attach to another switch which makes the host in not reachable state.

5.5 Results

We perform simulations under different conditions. To simulate real environment, we generate iperf packets from MN to CN throughout the process in background. For verification of network connectivity when a host moves from one switch to another switch we triggered ping. Based on the results of ping we can verify that even after host movement every other host will be communicated. For measuring handover latency, I used three different cases with host movement.

First case, host MN will move from one switch to another switch in a different and starts communicating with CN. Second case, host MN will move from switch in domain 1 to domain 2 and then to domain 3 which is not a same domain as CN and starts communicating with CN. Third case, host MN will move from domain 1 to domain 2 and then to domain 3 and then to domain 4 which is the same domain as CN.

Time taken for handover and start communicating with CN in all three cases is mentioned in the table below.

Table 2: Handover Latency under Different Background Traffic

Mobility / Approaches	Case 1	Case 2	Case 3
Vertical approach	7101000 ns	11132000 ns	19294000 ns

For the next evaluation, we measured the packet loss between the host MN and CN when the host is moving from one switch to another. For measuring the packet loss, we initiated the ping session between MN and CN in MN xterm window. After that we detached the host from the network and attached it to another network. The packets were dropped for a very small period. In [2] [3] the authors discussed in the results that they have a very low packet loss rate. Our packet loss rates are also like them. The three cases mentioned above matches with the real-time scenario regarding the mobility frequency. The results are shown in Figure 12.

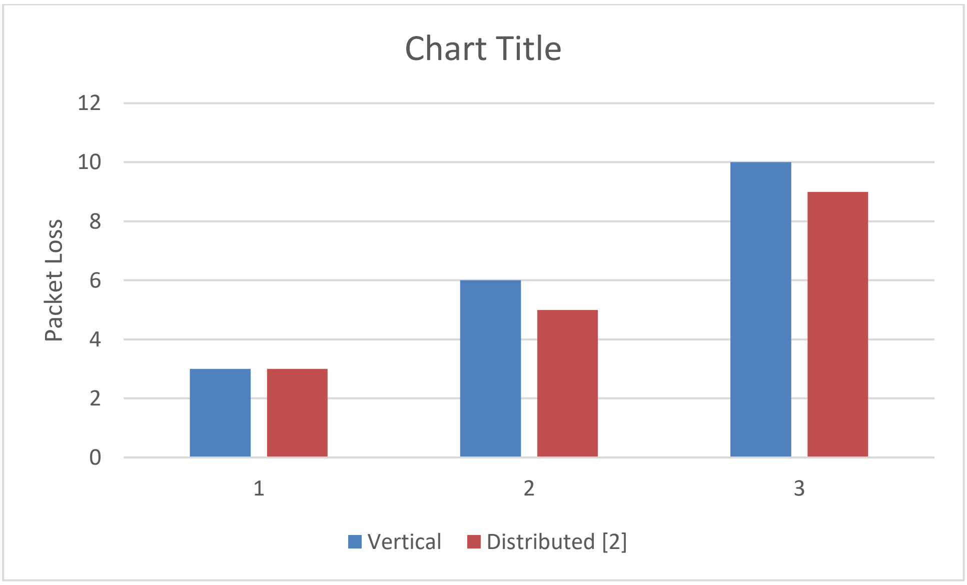


Figure 12: Packet Loss when host moves in Vertical and distributed approach

6. Simulation Evaluation

The handover latency for the proposed architecture is carried out using three different cases. As shown in Table 2, for vertical architecture, the handover latency for a host MN having different mobility frequencies is small levels. And under the same background traffic, in comparison with the distributed topology architecture, vertical topology architecture also produces the same result. The results indicate that the time taken for handover in centralized architecture is shorter. By adding Root controller in the topology, we can reduce the inter-domain communication cost.

Packet loss per host of different mobile frequency of both the topologies are compared. The results of the distributed approach are given in [2][3]. As shown in Figure 12, for the same levels of mobile frequencies in both topologies, we can see the distributed controller topology has less packet loss value over centralized controller topology. And with growing number of users, distributed topology and if we calculate the mobility frequency of each user the results show that both distributed and centralized topology has minimal difference in packet loss. But the distributed controller topology still outperforms the centralized controller topology under each user category. The results indicate that the distributed controller topology can provide better data service for each user under the same condition.

The proposed centralized controller architecture partly satisfies the goal considered by reducing the communication cost between the controllers. But it may potentially increase the packet loss because when the root controller is slow in some situations or in the cases like the root controller is not working all the sub-controllers will not work properly. So, it is a better opinion that for a network administrator who has high capacity controllers can use that it store all the mapping information and get benefit of not waiting for root controllers to take necessary action.

7. Conclusion

In this project, we have done a comparative analysis of centralized SDN topology and distributed SDN controller topology [2] [3] in packet loss. We proposed the centralized SDN topology and a binding cache solution for avoiding triangle routing and to minimize the cost of inter-controller communication. Simulations show that the distributed SDN controller has better handling packet loss than the proposed centralized topology. But it partly satisfies the goal by reducing the communication cost between controllers. In future work, we are planning on prototyping our proposal based on real SDN devices and network environment.

8. References

[1] C. Perkins. IP Mobility Support for IPv4. RFC 3344, 2002.

[2] Y. Wang and J. Bi, "A solution for IP mobility support in software defined networks," *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Shanghai, 2014, pp. 1-8.

[3] Wang, Y., Bi, J. & Zhang, “Design and Implementation of a Software-Defined Mobility Architecture for IP Networks”, K. Mobile Netw Appl (2015) 20: 40. doi:10.1007/s11036-015-0579-2

[4] McKeown N, Anderson T, Balakrishnan H et al (2008) OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 38(2):69–74

[5] "Software-Defined Networking (SDN) Definition - Open Networking Foundation". Opennetworking.org. 2017, “https://www.opennetworking.org/sdn-resources/sdn-definition”

[6] “Introduction to Mobile IP”, Cisco, 2001, “http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/mobile_ip/mobil_ip.pdf”

[7] “Software-Defined Networking: The New Norm for Networks”, Opennetworking.org. 2012, “https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf”

[8] Yap KK, Huang TY, Kobayashi M, Chan M, Sherwood R, Parulkar G, McKeown N (2009) “Lossless Handover with n-casting between WiFi-WiMAX on OpenRoads”, ACMMobicom (Demo) 12(3):40–52

[9] Yap KK, Kobayashi M, Underhill D, Seetharaman S, Kazemian P,McKeown N (2009) “The stanford openroads deployment.” In: Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization (pp. 59–66).

[10] P. Pupatwibul, A. Banjar, AAL Sabbagh and R. Braun. Developing an Application Based on OpenFlow to Enhance Mobile IP Networks. Local Computer Networks (LCN) 2013 Workshop on Wireless Local Networks, 2013.

[11] Wolfgang Braun, Michael Menth. “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices”, Future Internet, 2014.

[12] POX Controller. http://www.noxrepo.org/pox/about-pox/

[13] Mininet: An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/

[14] POX wiki. https://openflow.stanford.edu/display/ONL/POX+Wiki/