DESIGN AND IMPLEMENTATION OF THE MOBILITYFIRST PROTOCOL STACK ON AN SDN PLATFORM USING IPV6

BY KRISHNAVENI BUJARANPALLY

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

DIPANKAR RAYCHAUDHURI

and approved by

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE THESIS

DESIGN AND IMPLEMENTATION OF THE MOBILITYFIRST PROTOCOL STACK ON AN SDN PLATFORM USING IPV6

by KRISHNAVENI BUJARANPALLY

Thesis Director: DIPANKAR RAYCHAUDHURI

This thesis presents the design, implementation and evaluation of the MobilityFirst (MF) future internet architecture (FIA) on a Software Defined Network platform using IPv6 header fields. Some of the key features of the MobilityFirst architecture include unique identifiers (GUID) which separate naming from addressing, hop by hop transport protocol, in network storage and computation, global name resolution service (GNRS) which contains mappings of GUIDs to Network Addresses, a storage aware intra domain routing protocol and an edge aware inter domain routing protocol. The prototype implementation has been done on an SDN platform using the Open source Floodlight controller and Open vSwitches supporting OpenFlow 1.3 protocol. The prototype encapsulates all the aforementioned features of the MobilityFirst architecture. In order for the OpenFlow protocol to effectively handle MobilityFirst packets, the host stack of the MobilityFirst architecture has been modified to map the source and the destination GUID in the routing header to the source and the destination address fields of the IPv6 header of the packet. The controller then leverages this mapping to identify the source and destination and installs flow rules in the switches to route packets for hosts in the same domain without the help of GNRS. For the source and the destination in different domains, the controller obtains the network address of the destination network from the GNRS and uses the nSPs to identify the appropriate border router to forward packets. Additional modules have been added to the Floodlight controller to analyze and set up flow rules for the MobilityFirst packets and to communicate with the GNRS. In order to implement the in-network storage capability for an SDN an API has been developed for the controller to interact with the storage device. The controller routes the chunks to the storage box when the destination is not reachable or when there is host mobility. The prototype has been evaluated for throughput and latency by conducting experiments on the ORBIT test bed. The processing time of the controller is in the order of few milliseconds and the throughput reaches 750 Mbps and 600 Mbps for 1 MB chunks when the source and the destination are in the same domain and different domains respectively. The latency introduced by the storage box depends on the time interval between the GNRS queries made by the controller and increases as the query interval increases.

Acknowledgements

I would like to express my deepest gratitude to my advisor Prof. Dipankar Raychaudhuri for his constant guidance and motivation. I consider myself fortunate to have such an understanding advisor. My sincere thanks to Dr. Hana Godrich and Dr. Naghmeh Karimi for being a part of the defense committee.

I would like to thank Ivan Seskar who has been of great help, for taking time to answer all the queries that I had during the initial stages of the project. My sincere thanks to Jiachen Chen for his valuable insights during the discussions we had throughout the course of this research work.

My deepest gratitude to Francesco Bronzino for his valuable inputs and for helping me with all the difficulties I had in understanding the nuances of the project. I would like to thank Suja Srinivasan, Shashikanth Rangan ,Pavan Kulkarni and the members of WINLAB for their support and motivation. I am forever grateful to my family for their unconditional love and support in whatever I have done throughout my life. Last but not the least I would like to thank my friends at Rutgers whom made my journey pleasant and wonderful.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The complexity and manageability of computer networks increases with an increase in the number of proprietary network equipment such as routers, switches, firewalls, load balancers and NAT boxes. With an increase in the number of applications being developed using new agile methodologies, the rate of change of configuration of the network devices also increases. This not only creates an additional burden on network administrators but also increases both the capital and operational costs of managing the network. The complexity of network management is augmented by a lack of a central point of configuration for these equipment. Software Defined networks solve the above problems by separating the control plane and the data plane of the network and allow the management of network services by abstracting the functionality of lower layers. Technologies such as Active Networks which relied on the computational information in the packets to change the operation of the underlying network laid the required foundation for Software Defined Networks. While traditional networks rely on routers for traffic redirection and to disseminate network reachability information, Software Defined networks have a centralized controller which defines how packets should be routed in the network. The controller has a centralized view of all the devices in the network and uses this information to install flow rules in the switches to facilitate the exchange of packets between hosts attached to these switches. By decoupling the control plane functionality from the data plane the network administrators can dynamically adjust network resources for changing traffic conditions. This centralized approach also facilitates easy introduction of new protocols which was one of the major goal of technologies such as Active Networks. Thus, SDN is one of the favorable platforms for the implementation of future internet architectures such as MobilityFirst.

MobilityFirst [6] is a clean state approach to designing a new future internet architecture, with major goals to achieve seamless mobility of end hosts, and to support multicast, multi homing and context aware applications. Some of the key features of MobilityFirst include GUIDs, which are globally unique identifiers to separate the names of the network objects from the addresses to facilitate seamless mobility, a massively scalable and a distributed global name resolution service (GNRS), generalized storage aware intra domain routing protocol, edge aware inter domain routing protocol, and an in-network compute and storage layer.

1.1 Problem Statement

The data transferred between two hosts in a MobilityFirst network is divided into chunks or protocol data units and the average size of a chunk is 1MB. Each chunk of data is further segmented into packets depending on the MTU of the downstream link. Every chunk has a unique chunk ID and the first packet of the chunk contains the MobilityFirst routing header which has the source and the destination GUID required for end point identification. The routers in the network aggregate all the packets belonging to a particular chunk based on their unique IDs. For source and destination in the same domain,the GSTAR routing module in the routers looks up the destination GUID in the routing header and then decides whether the chunk should be forwarded to the next hop router or a GNRS look up is required or whether the chunk should be stored locally because of a poor downstream link quality. When the source and the destination belong to different domains, the EIR routing module uses the Network Address obtained from the GNRS and the information obtained from the Network state packets to route the packets to the appropriate network.

An SDN implementation of the MobilityFirst architecture should support all the above mentioned features. Open vSwitches which run the OpenFlow protocol do not support the functionality of aggregating and splitting the data into chunks. Hence, the flow rules for MF data stream in the SDN network will be installed on a per packet basis instead of a per chunk basis. Also, the OpenFlow protocol limits the matching fields to layer 2 and layer 3 for installing rules in the switches.

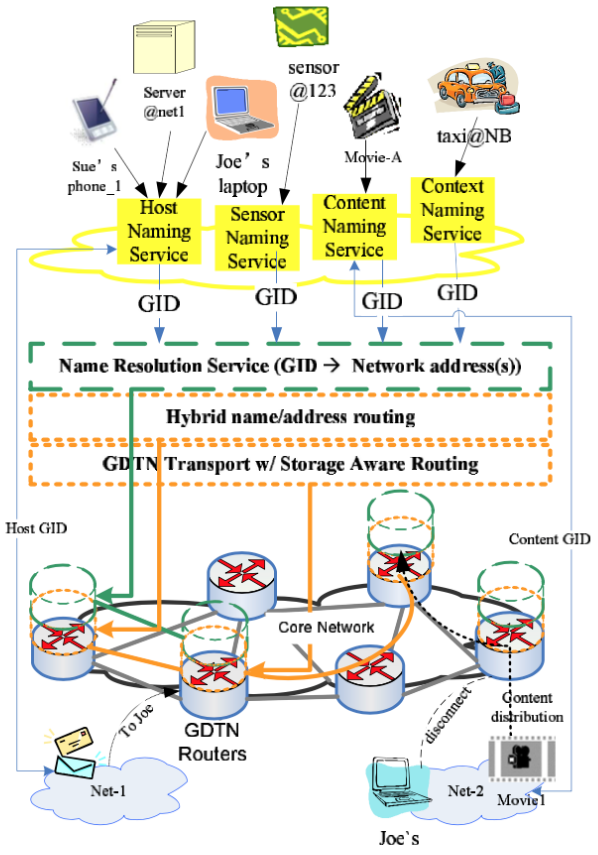

Figure 1.1: MobilityFirst Architecture protocol stack [1]

As a result MobilityFirst traffic should also be handled using either L2 or L3 flows. In order for the OpenFlow protocol to handle MobilityFirst packets the host stack in the MobilityFirst client has been modified to map the source and the destination GUID fields in the routing header to the IPv6 source and the destination fields in the layer 3 header. The controller then uses the IPv6 source and the destination address fields as matching fields to forward packets in the network and also to identify the end points.

Also, Open vSwitches do not have any local storage capability to provide the in network storage and compute functionality. In order to provide in network storage for late binding scenarios, an additional storage device can be attached to one of the ports of the switch. This storage device will be treated as one of the hosts in the SDN network by the controller. The storage device has a unique GUID and the controller identifies the storage device by the link probe messages sent as part of it's initial discovery protocol.

An API has been designed for the controller to interact with the storage device. The controller uses the API to send the packets to the attached storage device during periods of disconnection and to retrieve the chunks based on the destination GUID.

1.2 Related Work

[7] presents how the concept of software defined networking can be used to test new clean state network architectures in real time conditions on production networks. In this current work we present the prototype of MobilityFirst, which is also a clean slate design of future internet architecture on an SDN platform. [8] proposes a design of application specific routing protocol in content delivery networks using OpenFlow switches. The authors of [9] present an OpenFlow based design using layer 2 VLAN tagging for the MobilityFirst architecture to bypass routing decisions in segments of network which are stable. This achieves better performance gains by reducing the delay induced by hop by hop protocol [10] and also achieves flow aggregation.

[11] proposes the OpenFlow protocol which allows researchers to run their experiments on heterogeneous networks in real time traffic conditions. Most of the present day open source SDN controllers use the OpenFlow protocol to communicate with the underlying switches in the network. The authors of [12] propose a fast Fully Polynomial Time Approximation Scheme to solve the controller's optimization problem for efficient traffic engineering in scenarios where SDN is incrementally deployed in the existing network. A new network control framework, Procera is proposed in [13] which helps network operators to specify network policies in a high level language and to easily troubleshoot performance issues in the network.

In [14] an SDN based solution for Information centric networks called CONET is presented and also extensions for the current OpenFlow protocol to support non IP packets are discussed. In [15] a centralized inter domain routing solution based on the principles of BGP is proposed and evaluated using the NOX controller and OpenFlow protocol.

1.3 Outline

Chapter 2 discusses the MobilityFirst internet architecture. An overview of the SDN architecture and OpenFlow protocol version 1.3 is presented in Chapter 3. The design details of the prototype are discussed in Chapter 4. The evaluation of the prototype in terms of latency and throughput and the experiments on the ORBIT test bed are discussed in Chapter 5. Conclusions of the evaluation and the scope for future work is presented in Chapter 6.

Chapter 2

Overview of the MobilityFirst Architecture

The excessive growth of mobile devices, network service virtualization and the advent of cloud computing are among the few driving forces for a new internet architecture. The traditional client server internet architecture is ill suited to the dynamic computing and storage needs of the present day applications. MobilityFirst is a clean state redesign of the internet architecture with a main aim to provide scalable mobility and a robust design to tolerate downstream wireless link quality fluctuations and disconnections. The current IP architecture uses unique IP addresses to identify both the host and it's location and is not well equipped to provide seamless mobility as there is no separation between the identity of the host and it's location. Though mobile IP may provide a feasible solution in the short run, there is a significant latency experienced at the receiver due to the triangle routing problem [16] depending on the distance between the home agent and foreign agent. Denial of service attacks, passive eavesdropping, insider attacks are also some of the security threats to mobile IP.

MobilityFirst alleviates the above mentioned problems by introducing globally unique and flat identifiers which decouple the identity of the host from its location. GUID's form the narrow waist of the MobilityFirst protocol stack [6]. In this chapter we will discuss the major design goals and the protocol components of the MobilityFirst architecture and in subsequent chapters we describe our design of the protocol stack on a Software Defined Network platform.

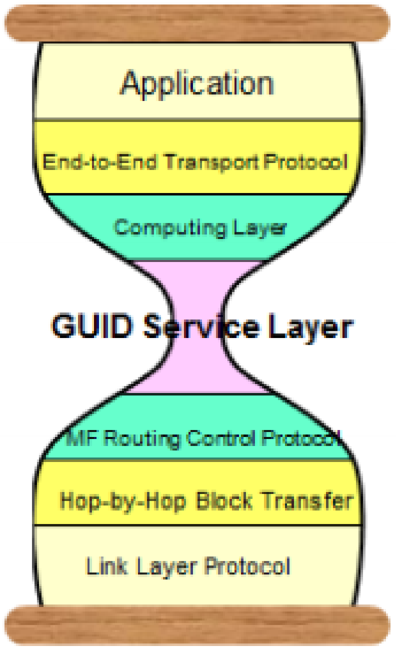
Application
End-to-End Transport Protocol
Computing Layer
GUID Service Layer
MF Routing Control Protocol
Hop-by-Hop Block Transfer
Link Layer Protocol

Figure 2.1: MobilityFirst Protocol Stack [2]

2.1 Naming and Name Resolution

To provide tolerance to disconnections and to ensure seamless mobility of the hosts, MobilityFirst separates the names and the location of the hosts by assigning a globally unique flat identifier called GUID to each network object in the network. These unique GUIDs are assigned to each network object by one of the many name certification services (NCS). These GUIDs are derived from public key thereby ensuring authentication and security services in the network. This self certifying property ensures that no external entity is needed for node authentication [17] and provides necessary security. Instead of using IP addresses, network services can now be invoked by using the source and the destination GUID and a service identifier (SID) to specify the mode of delivery such as unicast or multicast. A hybrid name/address based scheme is used for routing in MobilityFirst and a distributed global name resolution service (GNRS) is employed to dynamically bind the GUID to a list of current network addresses. The complete design specifications of GNRS can be found in [18] and [19]. When a list of Network addresses are available for a GUID, the router needs to only look up the NA routing table and this is referred as fast path forwarding. Routers also have the option of slow path forwarding where the GUID is rebinded to a new network address in case of disconnections or mobility, by periodically querying the GNRS.

GUID
List of Network Addresses the GUID is currently located in

Figure 2.2: GNRS Mapping

2.2 In Network storage and Compute Service

Every router in the MobilityFirst network also has a local storage cache to temporarily store chunks in case of disconnections or poor downstream link quality. This prevents the re transmission of the missing chunks by the sending host and reduces the total latency in receiving data. Older chunks in the cache are evicted on a First In First Out policy. The downstream path quality is constantly monitored and the chunks are forwarded when the link quality goes above a particular threshold. This threshold value is the weighted average of the long term link quality values. This storage and computing layer at the routers can be invoked for certain SIDs to support content caching, context aware message delivery or location aware routing.

2.3 Generalized Storage Aware Routing (G-STAR)

The distinguishing feature of the intra domain routing protocol of the MobilityFirst architecture is to temporarily store data in the network routers in the case of failure of wireless links [1]. The routing decision of whether to store or forward the data is a network layer decision and hence every router takes a per hop decision of either storing the content in case of poor downstream link quality, forwarding to the next hop when the conditions are favorable or caching the content for future use in content delivery networks. Since MobilityFirst uses a hybrid name and routing scheme the routing header of the each protocol data unit (PDU) or a chunk consists of GUID information, NA information and service tags.

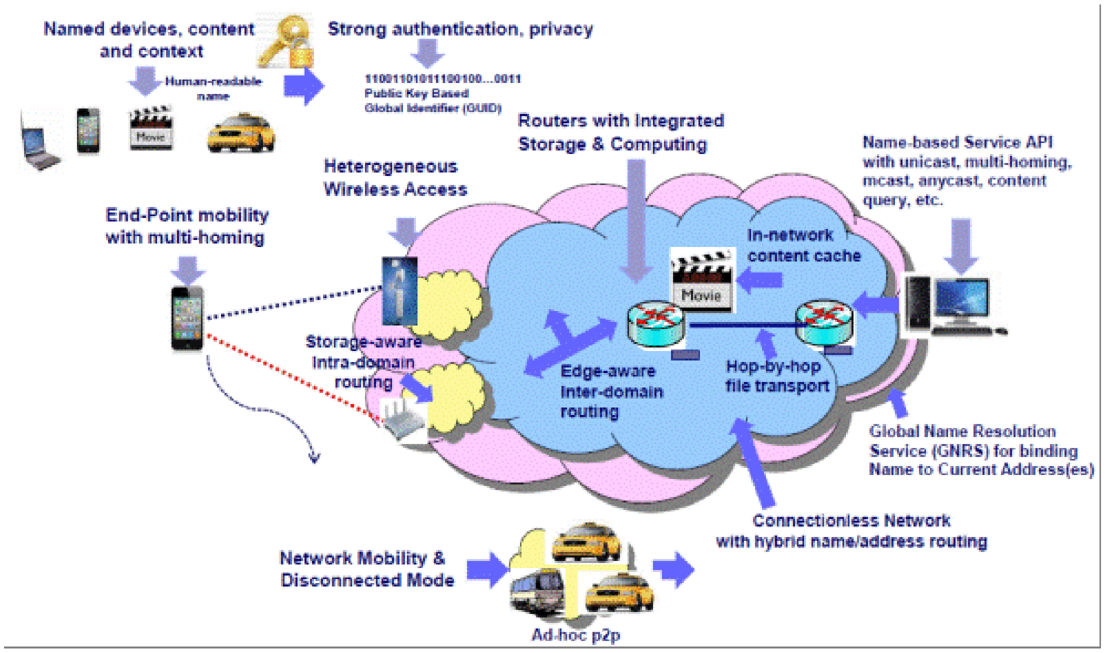
Named devices, content and context
Strong authentication, privacy
Human-readable name
11001101011100100...0011 Public Key Based Global Identifier (GUID)
Movie
Routers with Integrated Storage & Computing
Heterogeneous Wireless Access
Name-based Service API with unicast, multi-homing, mcast, anycast, content query, etc.
End-Point mobility with multi-homing
In-network content cache
Storage-aware Intra-domain routing
Edge-aware Inter-domain routing
Hop-by-hop file transport
Global Name Resolution Service (GNRS) for binding Name to Current Address(es)
Connectionless Network with hybrid name/address routing
Network Mobility & Disconnected Mode
Ad-hoc p2p

Figure 2.3: Components of MobilityFirst Internet Architecture [3]

The next hop router is determined based on the destination GUID. All the nodes in the network broadcast time sensitive fine grained link quality information about their one hop neighbors in the form of link state advertisements. These LSA messages contain short term (SETT) and long term estimation time (LETT) to their one hop neighbors. An intra-partition graph is then formed by calculating the link quality of all the nodes by using Dijkstras algorithm. Every node has complete information about the link quality to all the other nodes in the current partition. The routers also maintain a history of all the previous link qualities and over a significant period of time compute a long term ETT value for a link. Every router makes a decision to forward a chunk to the next hop if the current short term link quality is much greater than the long term link quality, and a decision to store the data if the current short term link quality is lower than the long term link quality. When the chunks are stored in the cache, a GNRS resolution for the Network address is made every five seconds and the chunks are redirected to the new location when a response is obtained from the GNRS.

2.4 Edge Aware Inter Domain Routing

The key component of the edge aware inter domain routing protocol is the abstraction of the Autonomous systems in terms of aggregated nodes (aNodes) and virtual links (vLinks) [20]. Every AS has the flexibility of dividing the routers into two or more groups of aNodes and has control over exposing the aNode's internal structure and state along with key properties such as bandwidth, latency and availability. The aNodes in the network can be connected by either a single or multi hop virtual links. EIR also allows networks to make forwarding decisions based on aggregated edge network properties by propagating coarse grained link level information about inter network links through the routing protocol. The border routers periodically disseminate messages containing information about the aNode-vLink connectivity and the internal structure of a network in the form of network state packets (nSPs). Internal to an AS all the routers exchange LSAs as part of the intra domain routing protocol of MobilityFirst and the border routers on receiving these LSAs and constructs an nSP by combing the enforced aNode topology of the AS and the export policy. The border routers telescopically flood nSPs received from other ASs to provide fast updates to nearby networks. In order to avoid routing failures to networks which are far away from the current networks, EIR has an additional feature of late binding of the GUID to NA during the packet transit. The border routers maintain multiple forwarding tables based on the vLink metrics obtained from nSPs and use the service identifier in the packet to choose the best one among the forwarding tables.

2.5 Hop by Hop Transport Protocol

In order to provide late binding functionality MobilityFirst uses a hop by hop transport protocol where the entire PDU or chunk is received before a decision of forwarding it to the next hop is made [10]. Every sender sends a control message called CSYN before sending data to the next hop. Every CSYN packet has a hop ID which is unique for every chunk and also contains information about the number of packets in the chunk. The receiver on receiving the CSYN packet allocates enough resources for the packet and replies with a CSYN-ACK packet with a bit map indicating the number of packets which are correctly received. This allows the sender to locally transmit the lost packets without unnecessary overhead due to re transmissions. The receiver aggregates all the packets with similar hop ID before looking up the routing header which is present in the first packet of the chunk to make a routing decision. In case of downstream disconnections every router stores the chunk locally and has a timer associated with every chunk. The local cache storage space is limited to a unique source and destination GUID pair and LRU policy is followed to evict older chunks to make room for newer incoming chunks. A GNRS query will be made to obtain a new NA after the timer associated with the storage chunk expires. In addition the transport protocol uses a back pressure scheme for congestion control and flow control.

Chapter 3

SDN and OpenFlow 1.3

The decision making functionality in most of the traditional internet architectures is implemented in dedicated network equipment which run proprietary software and are controlled through device specific command line interfaces and GUIs. The lack of a central point of control for these equipment makes managing heterogeneous networks error prone and cumbersome [21]. Also, no device in the network has a centralized view of the network to effectively react to dynamically changing traffic conditions. SDN on the other hand is a network architecture paradigm that separates the data, control and application planes [4] in the network. The intelligence of the device which consists of the packet forwarding logic is separated from the packet forwarding path and moved to a centralized controller.

The control plane in an SDN is a logically centralized entity with a view of the whole network and the data plane consists of simple forwarding elements such as switches. The control plane manipulates the behavior of the network by modifying the flow tables in the switches through the control channel. The control channel has a protocol that translates the controller logic into instruction sets that the switches can follow and vice-versa. This approach allows the network control logic to be directly programmable and the underlying infrastructure to be abstracted for applications and network services.

3.1 SDN Architecture

Figure 3.1 gives a high level overview of the SDN architecture [22] and the corresponding components.

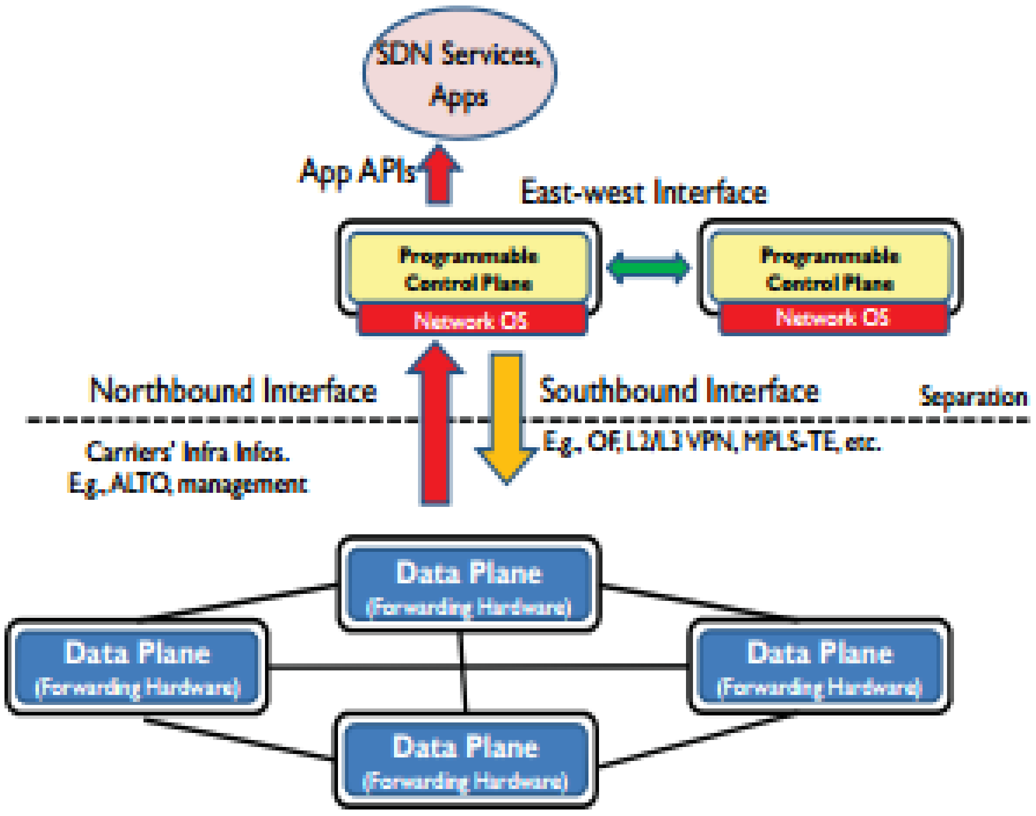
SDN Services, Apps
App APIs
East-west Interface
Programmable Control Plane
Network OS
Programmable Control Plane
Network OS
Northbound Interface
Southbound Interface
Separation
Carriers' Infra Infos. E.g., ALTO, management
E.g., OF, L2/L3 VPN, MPLS-TE, etc.
Data Plane (Forwarding Hardware)
Data Plane (Forwarding Hardware)
Data Plane (Forwarding Hardware)
Data Plane (Forwarding Hardware)

Figure 3.1: SDN architecture [4]

3.1.1 SDN Controller

The SDN controller is a centralized entity and the controller infrastructure consists of a North Bound API, south bound API and controller logic. The controller logic translates the network requirements of the applications into rules that can be installed in the underlying infrastructure through the south bound API. The north bound API is used by the application to interact with the controller logic and vice versa. The controller is also reponsible for providing the application with an abstracted view of the network. The flow rules can be installed in the underlying switches reactively or proactively based on the implementation.

3.1.2 SDN Datapath

The SDN datapath consists of networking devices such as simple switches whose functions include data processing and forwarding. The SDN controller interacts with the underlying switches through one of the widely used South Bound protocol, Open Flow.

3.1.3 SDN Applications

SDN applications are user defined high level programs that communicate the desired network behavior and requirements to the SDN controller via a North Bound Interface.

3.1.4 North Bound and South Bound API's

The south bound API or the control to data plane interface (CDPI) is the interface between the SDN controller and the underlying network switches and provides event notifications for the switches, advertises the switches capabilities to the controller, and programmatically installs flow rules in the switches. North Bound Interface is the interface between the controller and the SDN application and provides an abstracted view of the network to the application and allows the application to define their network requirements.

3.1.5 East Bound Interface

East bound interface is the interface between controllers overseeing different domains and is responsible for exchanging messages between the controller to provide services like inter-domain routing, intra domain routing, inter operability and scalability.

3.2 OpenFlow

OpenFlow is one of the south bound API which is widely adopted by the Software Defined Networking community. OpenFlow is not only a protocol defining the communication between the SDN controller and the underlying network infrastructure, it is also a specification of the logical architecture of the network switch. OpenFlow also allows researchers to try new experimental protocols by partitioning the traffic into production and research flows in the switches. The SDN controller communicates with the dedicated OpenFlow switch or OpenFlow compatible switches through a secure socket layer (SSL). The OpenFlow specification v1.3 defines three types of tables in the switches.

1. Flow Table: Flow table specifies header fields to be used for matching and a list of actions to be performed on incoming packets with similar header fields. Every flow table entry consists of match fields, priority, counters, instructions, timeout and cookies. OpenFlow specification defines four type of instructions that can be executed when the packet matches the entry. The instructions include directing packet through the pipeline to the next flow table, performing specified action on the packet like forwarding or dropping, updating meta data or action set of the packet.

2. Group Table: Entries in the group table contain a list of action buckets depending on the group type. The actions in one or more than one bucket can be applied to the packets sent to the group.

3. Meter Table: A meter table contains per flow metrics which enable OpenFlow to perform simple to complex QoS operations.

3.2.1 Flow Table Pipeline

OpenFlow switches may contain multiple flow tables and every table consists of multiple entries. The flow tables are sequentially numbered starting from zero. The incoming packet is always matched first against entries of Flow table 0. If a flow entry is found in table zero then the instruction set corresponding to that flow entry is executed. The instruction set may also direct the packet to another flow table whose number is higher than the current flow table. If the packet do not match against any entry in the flow table then the packet is dropped. If the flow table contains a table miss entry then the actions might include dropping the packet or sending it to the controller for further processing. The table miss entry does not exist by default in the flow table and may be added or removed by the controller at any time. Figure 3.2 shows the processing of an incoming packet by the pipeline.

3.2.2 OpenFlow Message Types

The protocol supports three types of messages, controller to switch, asynchronous and symmetric.

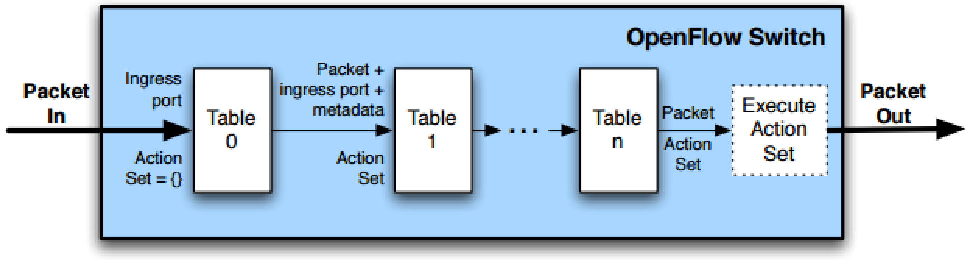

Figure 3.2: OpenFlow Processing Pipeline [5]

1. Controller to Switch: The controller to switch messages are the messages initiated by the controller and do not require a response from the switch. The controller sends these messages to query the switch for it's configuration parameters and features. The controller may also send modify state messages to add, delete and modify flow table entries in the flow table of the switch.

2. Asynchronous: Asynchronous messages are sent from the switch to the controller without any request from the controller. These messages are sent by the switches in case of events like Packet-ins, Flow removal, error or change in the switch state.

3. Symmetric: Symmetric messages are sent by the controller and the switches without either of them requesting for it. They include keep alive messages between the controller and the switch like Hello and Echo messages.

3.2.3 Message Handling

The OpenFlow protocol does not provide explicit acknowledgments on message delivery. In case of processing failures by the switch, OpenFlow protocol specifies that the switch should send back an error message to the controller.The ordering of the message processing can be maintained at the switches by the use of barrier messages. Messages are not reordered across a barrier message and the barrier message must be processed only when all the prior messages have been processed.

3.3 Floodlight

Floodlight is a combination of the Floodlight controller and applications built on top of it. The applications can be either standalone Java applications or applications built using the Floodlight JAVA API. Floodlight can be configured to load only the required application modules on startup by making changes to two property files.The Floodlight controller uses OpenFlow protocol as a South bound API to interact with the underlying switches. Floodlight uses the OpenFlowJ-Loxigen library to implement the OpenFlow protocol. The OpenFlowJ-Loxigen library has one common version-agnostic API for OpenFlow versions from 1.0 to 1.3. Figure 3.3 shows the architecture of Floodlight.

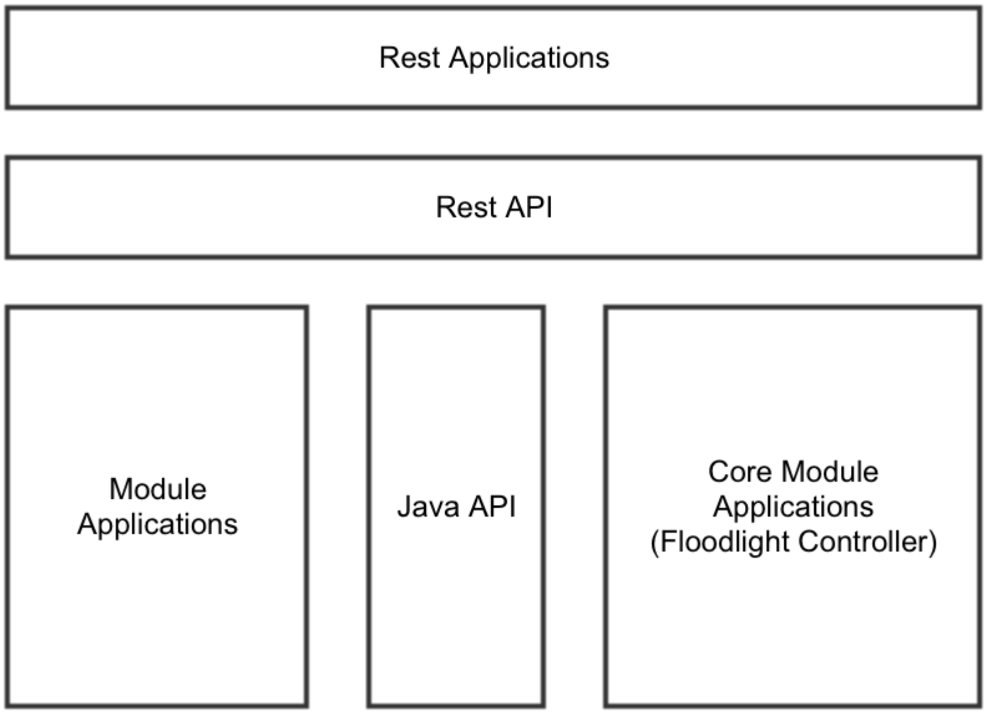

Figure 3.3: Floodlight architecture

Chapter 4

Design of the SDN prototype for MobilityFirst

This chapter presents the design of the MobilityFirst architecture on an SDN platform. When SDN is incrementally deployed in an existing MobilityFirst network there will be an interconnection of traditional MF networks and SDN based MF networks. Every domain or AS can either be a traditional MF network where the hosts run the MobilityFirst host protocol stack and are connected by click modular software routers or a software defined network composed of OpenFlow capable switches controlled by a centralized controller. As the OpenFlow capable switches do not have in built storage capacity additional storage boxes attached to these switches on one of the ports will also be a part of the SDN domain. For the SDN to have all the functionalities of the MobilityFirst architecture described in Chapter 2, the controller needs to perform the following functions.

1. Keep track of the GUID's of the border routers, hosts and storage boxes which are a part of the SDN. In addition to the GUID's the controller should also maintain the network attachment points of these elements.

2. Update the network addresses of the hosts in the SDN domain by sending insert messages to the GNRS.

3. Aggregate link probe packets from all the border routers and send them to all the hosts in the network. This should be done when there is more than one designated border router in the network. The controller should also redirect the link probe ACK packets from the hosts to the border routers.

4. For communication between hosts in the same SDN domain, the controller should install flow rules in the underlying switches. These rules should be based on the

GUID's and their corresponding network attachment points learnt during the host discovery phase.

5. For destinations not belonging to the SDN domain, the controller should send a GNRS look up message to determine the NA and use the information aggregated from the nSP's sent by the border routers to install flow rules in the switches to forward packets to the appropriate border router.

6. The controller should also install flow rules to send packets to the storage box when required and retrieve the packets form the storage box based on GUID when the NA of the destination is known.

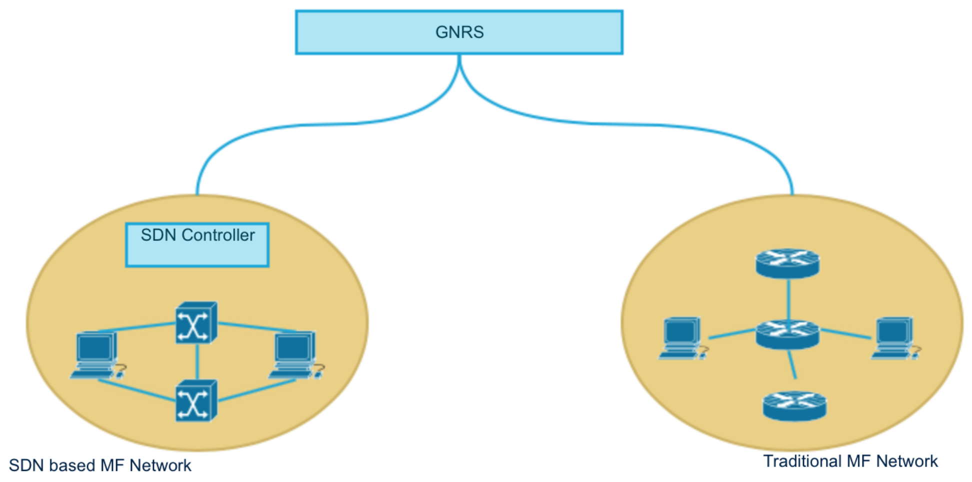

Figure 4.1: Coexistence of the traditional and SDN based MF networks

4.1 Discovery Phase

The traditional MF click modular software router broadcasts link probe messages on all the interfaces to learn about the one hop routers and hosts attached to it. The controller learns the GUID's of the border routers in the SDN domain from these link probe messages. These GUID's are then mapped to the MAC and IPv6 address of the router. Every border router is uniquely identified by the tuple <GUID, MAC Address, IPv6 Address >. On receiving the link probe messages on an interface the MF host replies with a link probe ACK and also learns the MAC and IPv6 address of the default router. The controller learns GUID's of the hosts from the link probe ACK messages. Again, every host in the SDN domain is uniquely identified by the tuple <GUID, MAC Address, IPv6 Address >. The storage boxes in the SDN domain also send a broadcast message containing their GUID as part of their discovery protocol. The controller learns the GUID, MAC Address and IPv6 Address of the storage box from these messages.

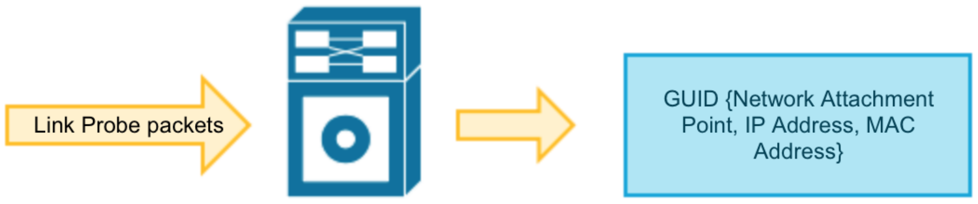

Figure 4.2: Mapping in the controller

4.2 Aggregation of Link Probe Packets

In scenarios where the SDN domain has more than one designated border router, the controller aggregates the link probe packets from all the border routers and sends out one link probe packet to all the hosts in the network with a default MAC and a default IP address. This ensures that the hosts do not switch their default router every time they receive a new link probe packet from the available border routers. When the hosts reply with a link probe ACK to the default gateway router the controller uses the information from the learning phase to install flow rules in the switches to modify the destination MAC and IPv6 address fields of the packet and forward it to all the border routers. Hence the neighbor tables of all the border routers contain information about all the hosts in the SDN domain.

4.3 Modifications to the host stack

The MobilityFirst host protocol stack has been updated to support IPv6 addressing capability. Modifications have been made to the host stack of MobilityFirst to map the source GUID and the destination GUID of the chunk which are present in the routing header of the first packet, to the source and the destination address fields of the IPv6 header of every CSYN and DATA packet. Also, the host keeps track of the source MAC and source IP fields of the CSYN packets that it receives. While sending a CSYN ACK packet the host uses the above fields as destination MAC and IP address. These modifications to the host protocol stack have been made to overcome the limitations of OpenFlow in supporting matching fields on non IP packets. The click router has also been updated to support IPv6 Addresses.

4.4 Intra domain Routing

As the centralized controller for the SDN domain is aware of the GUID's of all the routers and hosts in the domain, no GNRS lookup query is required to install flow rules in the underlying switches. On receiving a CSYN packet, the controller demaps the source and the destination GUID from the IPv6 address fields. The controller then obtains the network attachment points for these devices learnt during the discovery phase. The routing module in the Floodlight controller gives the Datapath ID's of the switches between the source and the destination and FLOWMOD messages are sent by the controller to all these switches. Better performance in terms of latency and throughput has been obtained when flow rules have been installed proactively for CSYN ACK packets using the information obtained from the CSYN packets.

4.5 Inter Domain Routing

For the source and the destination in different domains, the controller needs to choose the appropriate border router to forward the packets. On receiving a CSYN Packet, the controller obtains the source and destination GUID from the IPv6 source and destination address fields. Upon learning that the destination is not a part of the SDN domain, the controller sends a Lookup request for the destination GUID to the GNRS, which replies with NA of the destination. The controller then uses the information obtained from the nSP's sent by the border routers to identify the appropriate border router to forward the packets. The Routing module in the Floodlight controller then gives the list of switches between the source and the appropriate border router and the controller sends FLOW MOD messages to all these switches.

4.6 Design of Storage Element

In case of host mobility there will be periods of disconnection which causes the mapping of GUID and NA in the GNRS to expire. In such scenarios, the controller does not obtain a NA mapping upon sending a GNRS Lookup request. The controller then installs flow rules in the underlying switches to forward such packets to the storage box attached to one of the switches. After installing the required flow rules, the controller periodically queries the GNRS to see if a NA can be obtained. The frequency of these periodic GNRS queries can be optimized to obtain better reaction time to disconnection.

The storage box aggregates chunks based on their destination GUID. Upon obtaining a NA from the GNRS response, the controller sends a Retrieve request containing the destination GUID to the storage box. The storage box acts like a MobilityFirst host except that it stores chunks instead of processing them. The storage box upon receiving the Retrieve request from the controller, sends a CSYN packet to the appropriate border router similar to a normal MobilityFirst host. After receiving a CSYN ACK from the border router acknowledging the receipt of all the packets of the chunk, the storage box clears the buffered space for the chunk.

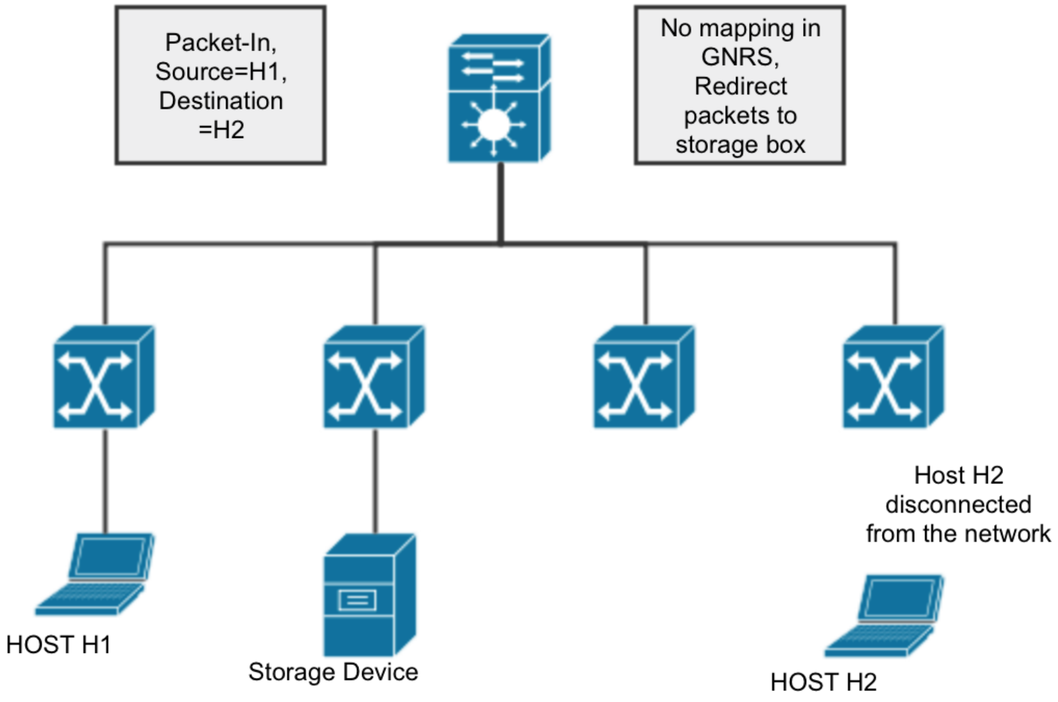
Packet-In, Source=H1, Destination =H2
No mapping in GNRS, Redirect packets to storage box
Host H2 disconnected from the network
HOST H1
Storage Device
HOST H2

Figure 4.3: Chunks being redirected to Storage Device when there is host mobility or disconnection

4.7 Floodlight Modules

1. Packet Classifier: The MobilityFirst forwarding module listens to the Packet-In messages sent by the switches. After classifying the packet as a MobilityFirst packet, the forwarding module sends them to the Packet Classifier module. The Packet classifier module then differentiates the MobilityFirst packet into Link Probes, Link Probe Acknowledgments, CSYN, CSYN Ack and Data Packets. This information is used by the Mapper module and the forwarding module.

2. GUID to MAC Mapper: This module maintains a mapping of the GUID's of objects to their respective MAC and IPv6 Addresses. It also maintains a list of GUID's of the hosts and routers in the SDN domain. In addition to this list, it also keeps track of the GUID's of the storage devices in the network.

3. GNRS Helper: This module is responsible for sending GNRS Lookup messages to get the NA mapping. It also sends Insert messages to the GNRS to update the NA of all the hosts in the SDN domain. This module also sends Retrieve Requests to the storage box when a mapping is obtained from the GNRS.

4. MobilityFirst Forwarding Module: The MobilityFirst forwarding module has functions to aggregate the link probe packets sent by the controller and functions to install flow rules in the appropriate switches for CSYN, CSYN ACK and Data packets.

Chapter 5

Evaluation and Validation

In this chapter we present the evaluation of the performance of the control plane and the data plane of the prototype. Experiments to evaluate the performance were conducted on the ORBIT test bed. The performance of the control plane has been evaluated on the basis of the processing time of the controller and data plane on the basis of the end to end throughput achieved in the prototype.

5.1 Control Plane

5.1.1 Processing time of the controller

The overall delay to set up flow rules in the switches is the sum of the time taken by the switches to send Packet-In messages to the controller, processing time of the controller and the time taken by the controller to actually install flow rules in the switches. The processing time of the controller further depends on whether the destination is a part of the SDN domain overseen by the controller or not. When the destination is not a part of the domain controlled by the controller, the processing time also includes the time taken by the controller to send a look up request to the GNRS and the response time of the GNRS. Table 5.1 summarizes the processing delay introduced by the controller in both the cases.

Case	Processing delay
Without GNRS query	8.375 msec
With GNRS query	22.44 msec

Table 5.1: Processing delay introduced by the controller

Table 5.2 shows the breakdown of the delay when the destination is in a different domain not controlled by the controller.

Case	Delay
GNRS Response time	9.05 msec
API Delay	3.04 msec

Table 5.2: Breakdown of the processing delay

5.2 Data Plane

In order to evaluate the data plane performance of the prototype, mfping and mfperf were used to calculate the end to end delay and throughput respectively. These experiments were conducted for different topologies where the source and the destination were in the same SDN domain as well as in different domains. The following sections of the chapter summarize the results obtained.

5.2.1 mfping

mfping is a network performance measurement tool which is used to measure the average round trip time between the MobilityFirst source and destination. The mfping tool sends 64 bytes of data encapsulated in MobilityFirst headers to verify network connectivity and to measure the average, maximum and the minimum round trip times between the source and the destination. We used mfping to measure the average RTT's for four different topologies. The topologies and the results obtained are summarized below.

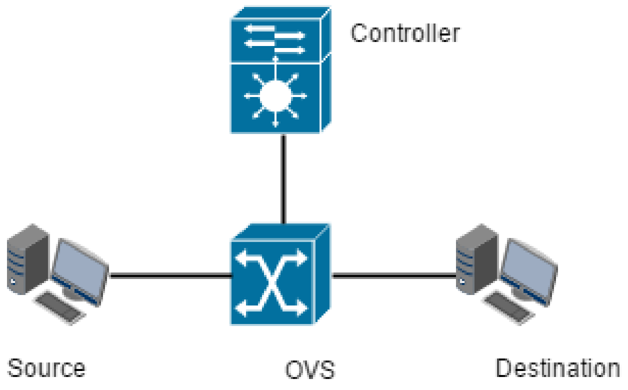

Figure 5.1: Topology with one OVS between source and destination

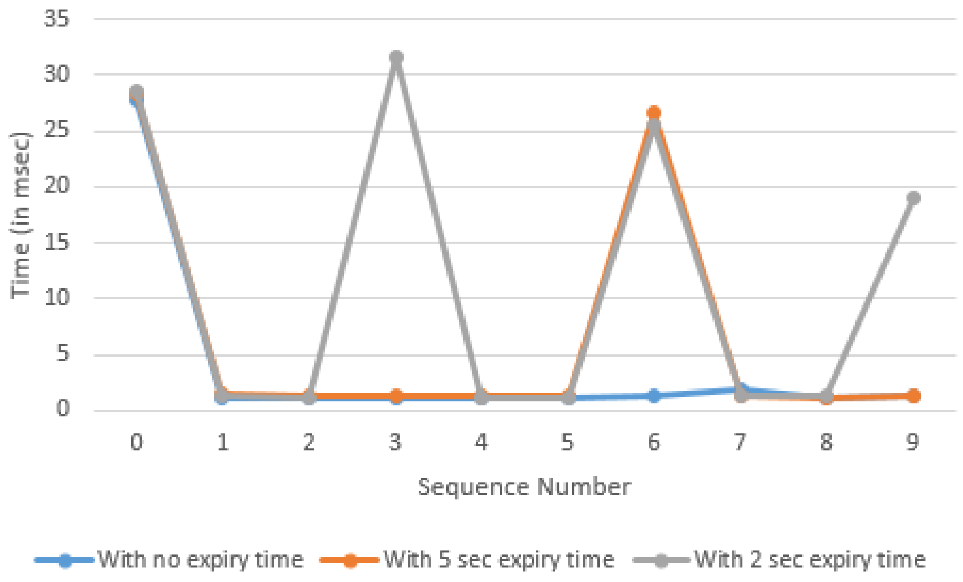

Figure 5.2: RTT Measurements for topology in Fig 5.1

Figure 5.2 summarizes the results obtained for topology where there is only one OVS between the source and the destination. To support host mobility the controller should clear the previously installed flow rules in the switches and install new rules depending on the new location of the destination. We carried out the experiments based on the assumptions that the destination moves every 2 and 5 seconds and the results are presented in the Figure 5.2. As can be seen from the above graph, when there is no host mobility only the first ping request has higher RTT as it includes the time for the switch to send a Packet-In message to the controller, processing time of the controller and the time to install flow rules in the switch. The subsequent ping packets have very low RTT's and the RTT includes only the network latency between the source and the destination. When the host moves every 5 seconds, the above mentioned process is repeated every 5 seconds and only the first and the sixth ping packets have higher RTT's. In similar lines, when the host moves every 2 seconds, the first, third, sixth and ninth ping packets have higher RTT values when compared to other ping packets.

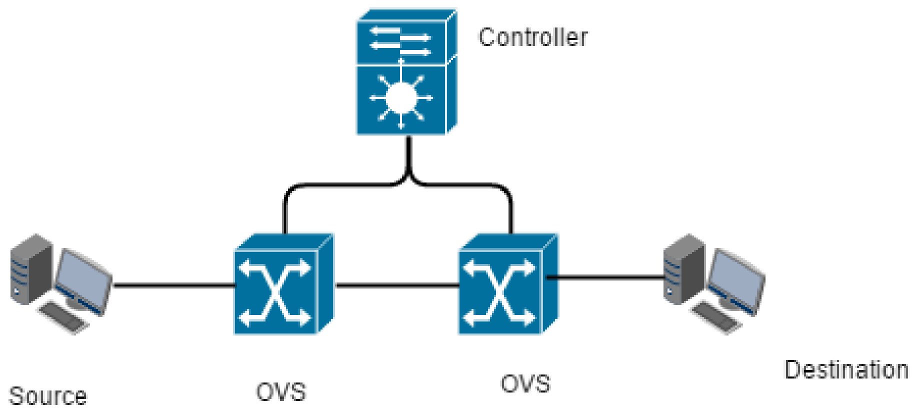

Figure 5.3: Topology with two Open vSwitch's between source and destination

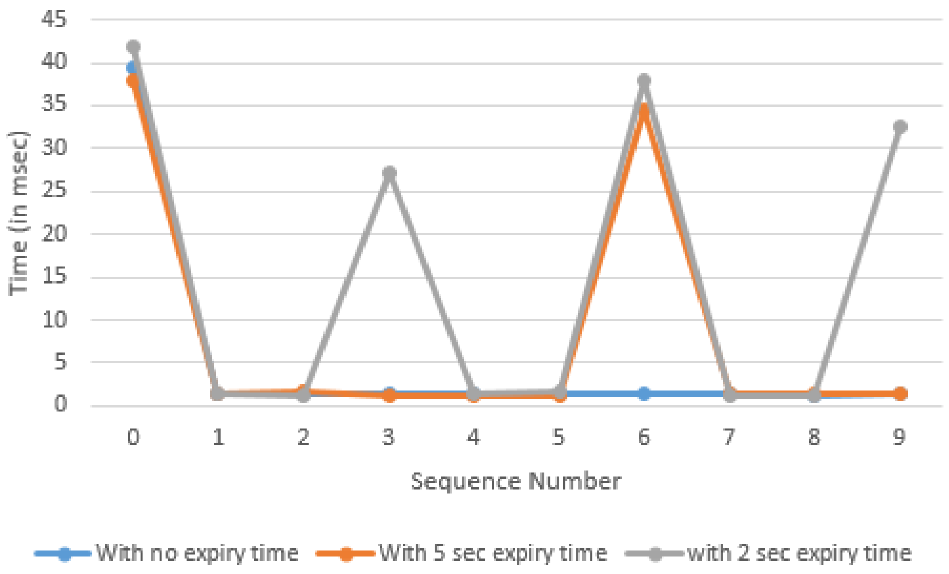

Figure 5.4: RTT Measurements for topology in Fig 5.3

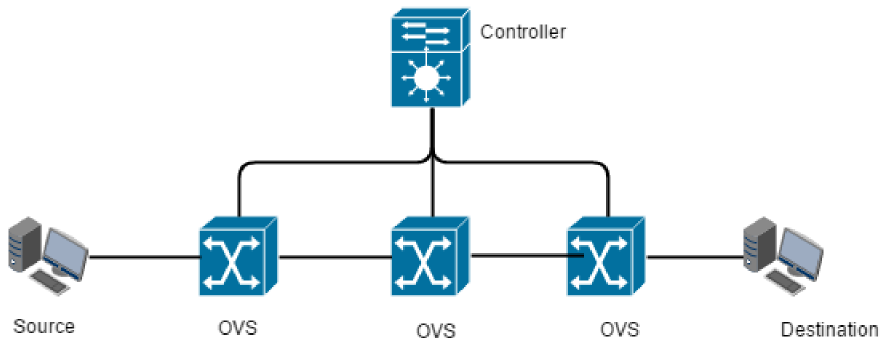

Figure 5.5: Topology with three Open vSwitch's between source and destination

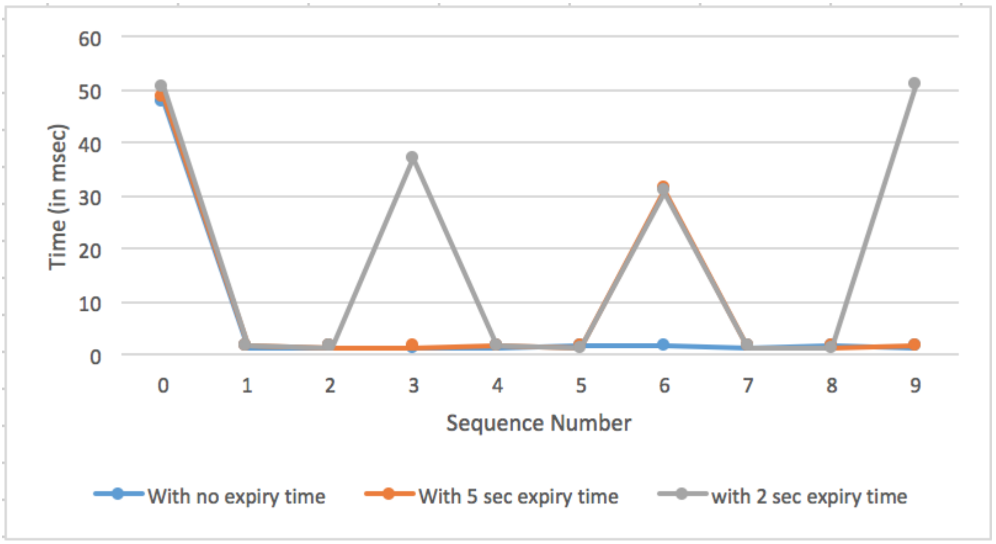

Figure 5.6: RTT Measurements for topology in Fig 5.5

Figures 5.4 and 5.6 summarize the results when there are two and three OVS's between the source and the destination respectively. As can be seen from figure the round trip times between the source and the destination are increasing as the number of switches between the source and the destination are increasing. This is because as the number of switches increase the time taken to install the flow rules in the switches increases.In addition, every switch has to now send a Packet-In message to the controller when there are no previously installed flow rules in the packet. To minimize this latency, we installed flow rules in all the switches between the source and the destination in both the directions upon receiving a CSYN packet from the first switch.

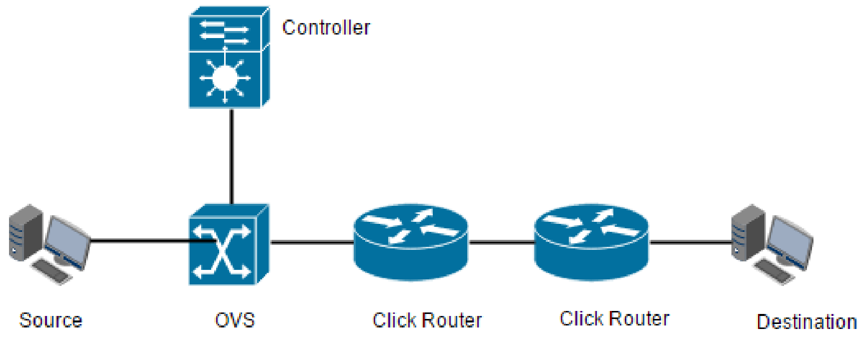

Figure 5.7: Topology where the source and destination belong to different domains

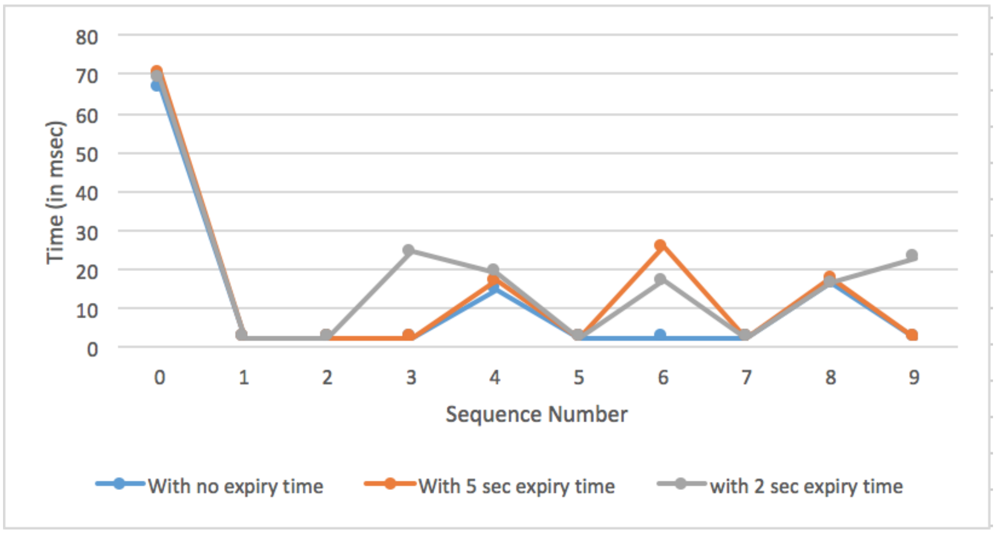

Figure 5.8: RTT Measurements for topology in Fig 5.7

Figure 5.8 summarizes the results when the source and the destination belong to different domains. As can be seen from the figure the RTT for the very first ping packet is higher than the RTT's obtained when the source and the destination are in the same domain. When the source and the destination belong to different domains, the processing time of the controller also includes the time taken to send a LookUp message to the GNRS and processing time of the GNRS. The RTT also includes the processing time of the Click routers between the source and the destination. When there is no host mobility only the very first ping packet has a higher RTT and the RTT's of the subsequent packets depend on the processing time of the click routers. The click routers send a GNRS Look up request every 5 seconds to support host mobility.

5.2.2 iperf

iperf is another network measurement tool which is used to measure the average throughput of a network. The MobilityFirst version of iperf uses User Datagram Protocol (UDP) packets and allows the user to specify the GUID's of the source and the destination and the chunk size of the Mobility First packets to be used. The throughput values obtained and the topologies are discussed below.

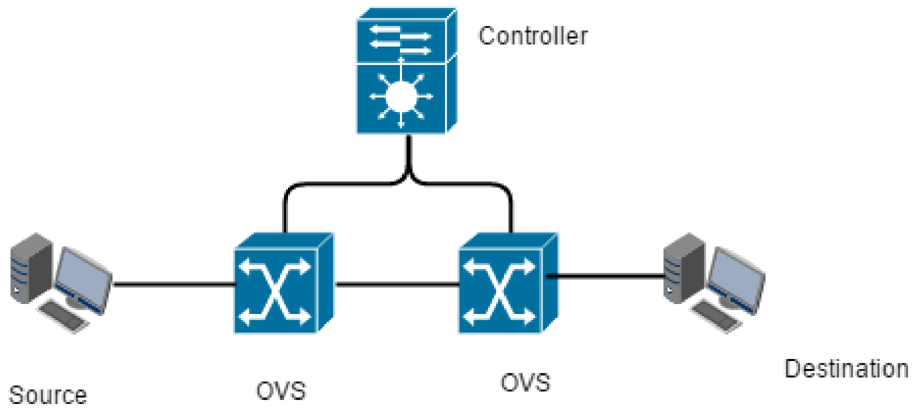

Figure 5.9: Topology with two Open vSwitches between source and destination

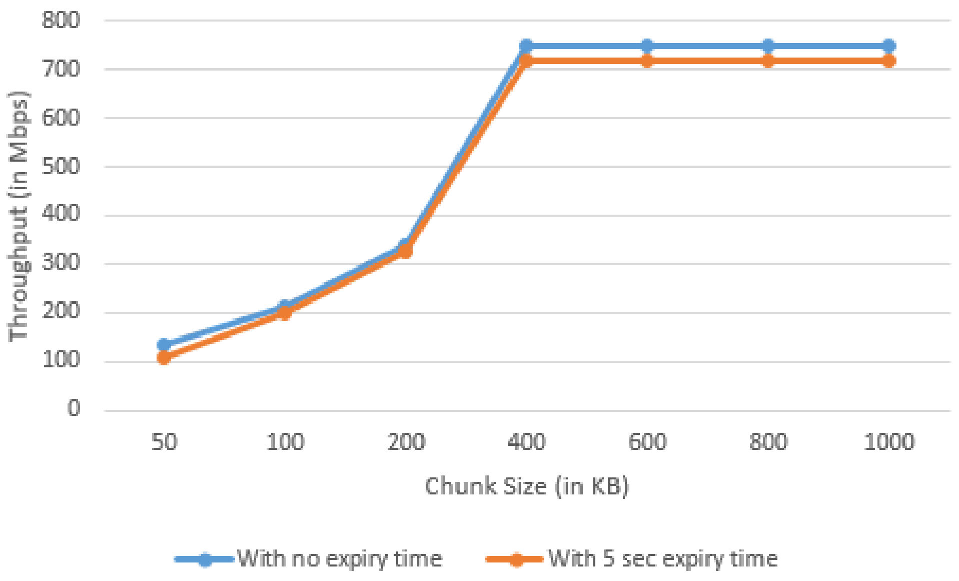

Figure 5.10: Chunk Size Vs Throughput when source and destination are in same domain

Figure 5.10 summarizes the throughput results when the source ad the destination are in the same domain. The chunk size has been varied between 50 KB-1 MB to observe the throughput. When there is no host mobility, the maximum throughput obtained is 750 Mbps for a chunk size of 1 MB. The throughput increases with increasing chunk sizes. For smaller chunk sizes there is an increased overhead due to the Mobility First transport and routing headers which results in smaller throughput values. As the chunk size increases, this overhead decreases relatively and hence an increase in the throughput. When the host is assumed to move every 5 seconds, the maximum throughput obtained is 720 Mbps for a chunk size of 1 MB. This decrease in the throughput is because the flow rules in the switches will expire every 5 seconds and the controller installs new rules every 5 seconds.

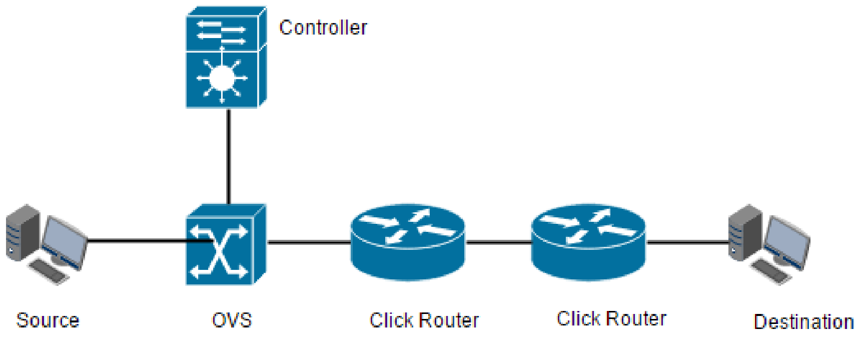

Figure 5.11: Topology when the source and destination belong to different domains

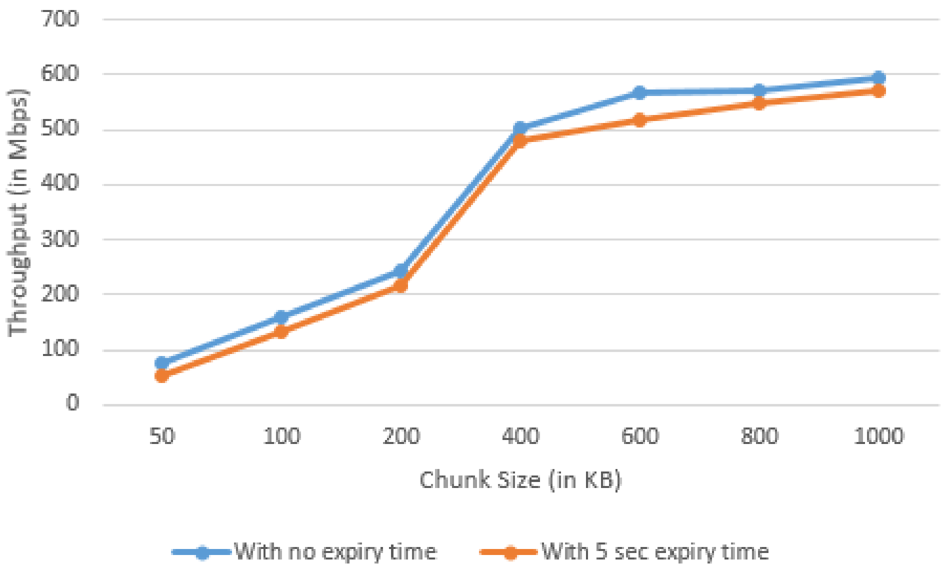

Figure 5.12: Chunk Size Vs Throughput when source and destination are in same domain

Figure 5.12 presents the results when the source and the destination are in different domains. The throughput values were recorded by varying the chunk size between 50

KB and 1 MB. The maximum throughput is obtained is 600 Mbps for a chunk size of 1 MB when there is no host mobility and 570 Mbps when the host is assumed to move every 5 seconds. When there are no click routers the floodlight controller installs flow rules in all the switches at once while click routers follow a hop by hop transport protocol to process chunks. And hence as can be seen from Figures 5.10 and 5.12, higher throughput is obtained when there are only OVS's than when there are OVS's and click routers exist together.

5.2.3 Latency measurements for attached storage devices

In order to evaluate the end to end latency introduced by the storage device, we used the topology show in Fig 5.13. The results are summarized in Fig.5.14.

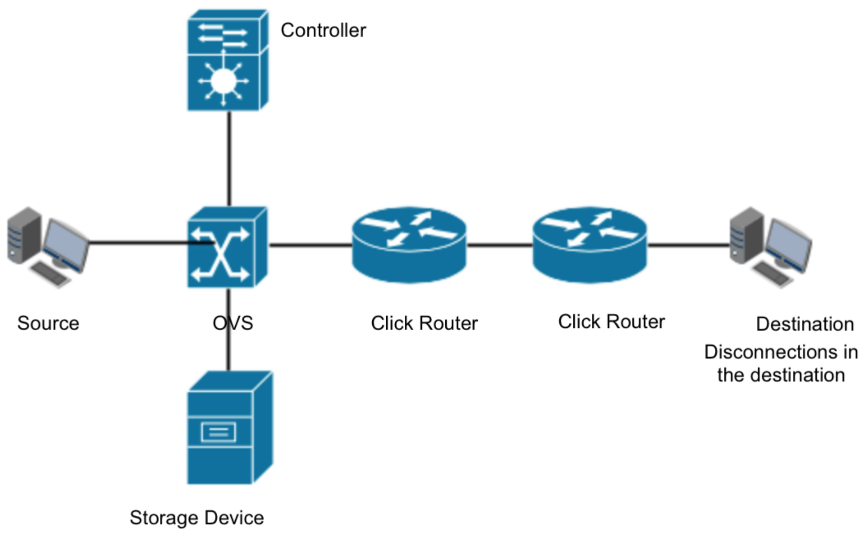

Figure 5.13: Topology for evaluating the end to end latency introduced by the storage device

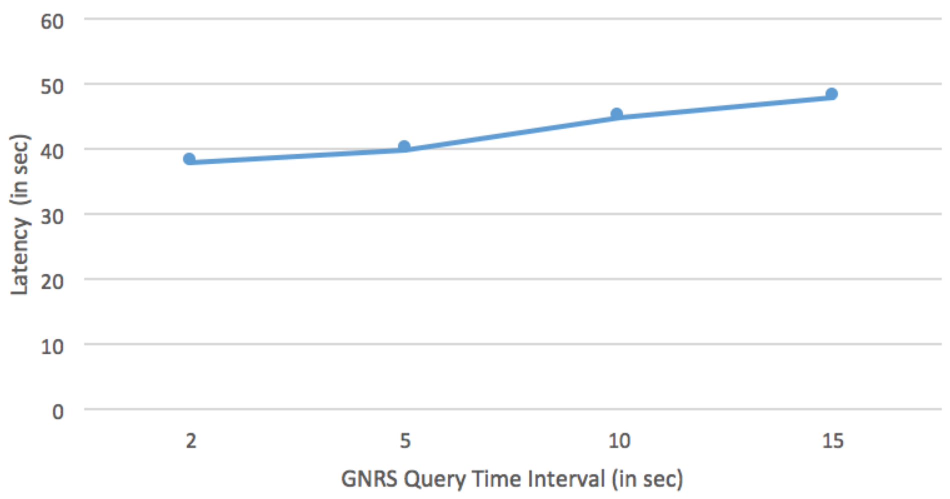

Figure 5.14: Variation of end to end latency with GNRS query interval

As the time interval between the GNRS queries made by the controller increases, the end to end latency increases. The end to end latency can be broken down into the sum of the network delay, processing delay of the controller, and processing time of the storage device and the time taken to get a response from the GNRS (depends on the re-connection time of the client). The experiment has been conducted for a chunk size of 1000 KB and the destination re-connection time is assumed to be 30 sec. The processing time of the controller which is the time taken to process a CSYN packet from the storage device and install flow rules in the underlying switches remains constant. The processing time of the storage device depends on the number of chunks stored for a particular destination GUID and the network delay depends on the underlying topology. The processing time of the storage box remains constant in this experiment for a chunk size of 1000 KB. As the interval between the GNRS queries increases, the time taken to get a response from the GNRS increases which increases the end to end latency.

Chapter 6
Conclusions and Future Work

The prototype shows that all the key features of the MobilityFirst architecture including in network storage capability can be implemented on an SDN platform. All the required services have been implemented by extending the modular structure of the Floodlight controller. The MobilityFirst host stack has been updated to support IPv6 addresses and the limitation of the OpenFlow protocol in supporting MobilityFirst fields has been overcome by mapping the source and the destination GUID to the IPv6 address fields. Evaluations have shown that the throughput of the prototype increases as the chunk size increases and reaches a maximum value when the chunk size is 1 MB. Higher throughput values have been observed when there were only Open vSwitches than when the network consisted of both OVS's and click routers. Also, having to do a GNRS lookup for inter domain routing added significant delay to the end to end latency. The processing time of the controller also increases when a GNRS lookup has to be done.

For future work, a protocol can be developed for inter domain controllers to communicate the information about hosts in their network to decrease the burden on GNRS. Any new service which will be added to the MobilityFirst architecture can be easily implemented owing to the modular structure of the controller.

References

[1] S. Nelson, G. Bhanage, and D. Raychaudhuri. Gstar: Generalized storage-aware routing for mobilityfirst in the future mobile internet. *Proceedings of the 6th ACM International Workshop on Mobility in the Evolving Internet Architecture(MobiArch)*, June 2011.

[2] Orbit website. Available: `http://mobilityfirst.orbit-lab.org/`.

[3] Mobilityfirst fia overview. Available: `http://mobilityfirst.winlab.rutgers.edu/`.

[4] Diego Kreutz, Fernando M.V. Ramos, Paulo Esteves Verissimo, Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive study. *Proceedings of the IEEE*, 103(1):14–76, January 2015.

[5] Openflow switch specification. *Open Networking Foundation*, June 2012.

[6] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani. Mobilityfirst: A robust and trustworthy mobilitycentric architecture for the future internet. *ACM SIGMobile Mobile Computing and Communication Review (MC2R)*, 16(4), October 2012.

[7] F. de Olivera Silva, J. de Souza Pereira, P. Rosa, and S. Kofuji. Enabling future internet architecture research and experimentation by using software defined networking. In *2012 European Workshop on Software Defined Networking (EWSDN)*, 2012.

[8] M. Othman and K. Okamura. Design and implementation of application based routing using openflow. In *Proceedings of the 5th International Conference on Future Internet Technologies (CFI'10)*, 2010.

[9] A. Lara, B. Ramamurthy, K. Nagaraja, A. Krishnamoorthy, and D. Raychaudhuri. Cut-through switching options in a mobilityfirst network with openflow. In *Proceedings of IEEE 7th International Conference on Advanced Networks and Telecommunication Systems (ANTS)*, 2013.

[10] K. Su, F. Bronzino, K.K Ramakrishnan, and D. Raychaudhuri. Mftp: A clean-slate transport protocol for the information centric mobilityfirst network. In *Proceedings of ICN conference*, 2015.

[11] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovations in computer networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April. 2008.

[12] Suman Agarwal, Murali Kodialam, and T.V Lakshman. Traffic engineering in software defined networks. *INFOCOMM, 2013 Proceedings IEEE*, pages 2211–2219, April. 2010.

[13] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, pages 114–119, February. 2013.

[14] Luca Veltri, Giacomo Morabito, Stefano Salsano, Nicola Blefari-Melazzi, and Andrea Detti. Supporting information centric functionality in software defined networks. *IEEE Internation Conference on Communications*, pages 6645–6650, June 2012.

[15] Ricardo Bennesby, Paulo Fonseca, Edjardo Mota, and Alexandre Passito. An inter-as routing component for software defined networks. *IEEE Network Operations and Management Symposium*, pages 138–145, April 2012.

[16] C.E Perkins. Mobile networking through mobile ip. *IEEE Internet Computing*, pages 58–69, 1998.

[17] D.G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol. *ACM SIGCOMM Computer Communication Review*, 38:338–350, 2008.

[18] A. Venkataramani, A. Sharma, X. Tie, H. Uppal, D. Westbrook, J. Kurose, and D. Raychaudhuri. Design requirements for a global name service for a mobility-centric, trustworthy internetwork. *IEEE COMSNETS 2013.*

[19] Yi. Hu, R.D. Yates, and D. Raychaudhuri. A hierarchically aggregated in-network global name resolution service for the mobile internet.

[20] S. Mukherjee, Shravan Sriram, Tam Vu, and D. Rauchaudhuri. Eir: Edge-aware inter-domain routing protocol for the future mobile internet.

[21] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software defined networking: Past, present and future of programmable networks. *IEEE Communications Surveys and Tutorials*, pages 1617–1634, February 2014.

[22] Myung-Ki Shin, Ki-Hyuk Nam, and Hyoung-Jun Kim. *2012 International Conference on ICT Convergence (ICTC).*