DESIGN AND IMPLEMENTATION OF A VIRTUAL NETWORKING FRAMEWORK FOR THE MOBILITYFIRST FUTURE INTERNET ARCHITECTURE

BY AISHWARYA BABU

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

Dr. Dipankar Raychaudhuri

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2015

ABSTRACT OF THE THESIS

DESIGN AND IMPLEMENTATION OF A VIRTUAL NETWORKING FRAMEWORK FOR THE MOBILITYFIRST FUTURE INTERNET ARCHITECTURE

by Aishwarya Babu

Thesis Director: Dr. Dipankar Raychaudhuri

This thesis presents the design and implementation of a virtual networking framework for the MobilityFirst (MF) future Internet architecture (FIA). The proposed design leverages the key feature of MobilityFirst by which all network-attached objects are given a globally unique identifier (GUID). Virtual networks (VNs) are supported in this architecture by assigning a virtual network GUID to each VN, and then using the global name resolution service (GNRS) to dynamically maintain topology information. Virtual networks in MobilityFirst also use an additional routing update message between nodes in the VN, enabling new capabilities such as application specific routing (ASR). The proposed virtual networking processing components are integrated into the prototype MobilityFirst software router which is based on the open source Click routing platform. Routing decisions are made using virtual routing tables maintained at each virtual router and these virtual tables are populated by VN control messages. Evaluation of the VN implementation is done using standard networking tools, ping and iperf, as well a proof of concept validation and demonstration. The ping and iperf experiments compare the performance of the virtual network over MF with the topology solely based on MF. A sample application specific routing based algorithm is implemented within the VN to visualize the effect of ASR using service anycast.

Acknowledgements

I would like to express my sincere thanks to my adviser Prof. Dipankar Raychaudhuri who has guided me through the course of this thesis. Despite his extremely busy schedule he has always made time to answer any questions that I might have. Thank you to Ivan Seskar for helping me understand many nuances of the project. I would like to thank Prof. Yanyong Zhang and Prof. Janne Lindqvist for being part of my defense committee. My deepest gratitude to Francesco Bronzino who has been a mentor throughout this work and has patiently helped me with difficulties that I had. I would like to thank everyone at WINLAB who have made this an enjoyable experience. Thank you to my friends who have been a family away from home. Lastly, I would like to thank my parents who helped make this journey possible.

Table of Contents

Chapter 1

Introduction

Over the past couple of decades the Internet has seen a tremendous increase in number of users and applications. From being merely a means of connecting computer systems, it has grown into being the biggest source of information [1]. The current IP-based internet architecture is designed to robustly support a relatively static topology and can, to a great extent, support mobility. However, in the recent past the number of mobile devices has started to increase that requires a more dynamic underlying protocol in order to serve high mobility which results in frequent disconnections, call drops, break in transfers, etc. The advent of Internet of Things, Wearables, Vehicular networks, etc. has and will only speed up the rate at which the count of wireless devices is growing. To be able sustain and support this growth in the number and variety of wireless devices, the current IP architecture with the connection oriented TCP model will not be sufficient.

To this end, the MobilityFirst (MF) Future Internet Architecture [2] has been proposed and its prototype is underway which puts mobility at the fore. MobilityFirst is a clean slate approach to handle this type of a network which predominantly consists of wireless/ mobile devices. In Section 2, we describe the design goals of the MobilityFirst FIA.

The current Internet architecture has successfully supported a multitude of applications provided by different providers. In order to allow this, the various providers need to be in agreement of the changes that need to be made to the architecture [3]. These changes, even when minor, need to be made with careful consideration so as not to have any adverse repurcussions. Thus, the accommodation of any new change into the Internet architecture is slow. An example is the deployment of Ipv6 which is taking quite a while to establish presence [4]. Introducing changes to the internet architecture would, in many cases,

involve upgrading the available infrastructure which comprises tens of millions of routers, servers and other network components [5]. This could imply a non-trivial cost of deployment for each update.
The slowing down of pace and difficulty to introduce changes, prevents the realization of many innovations into the Internet architecture. As a result, virtualization has become the means to encourage experimentation and innovations in networking. Virtual networking provides the basis for flexible topology definition, resource isolation and introduction of custom routing algorithms. In this project we design and implement a virtual network architecture to be deployed on a physical substrate running the MobilityFirst stack.

MobilityFirst's design inherently allows for integrated/ native support for virtual network routing with its named object architecture. Every MF network object, be it router, end host or even an entire network can be recognized by what is called a GUID (Globally Unique Identifier) in MF. Being able to uniquely reference each virtual network and its virtual members helps achieve isolation with much ease.

A use-case to exemplify the advantages of this virtual network design is a concept called Application Specific Routing(ASR). ASR is a routing mechanism wherein routing decisions are taken based on a parameter which combines the network layer's metric along with a metric specified by the application. For example, we can combine the latency measured by the routing layer and the compute load measured at the application layer. This gives the service providers the flexibility to incorporate parameters which allow for utilizing the intelligence about Layer 4 - 7 [6]. Some use-cases can exemplify the advantages of having ASR. A gaming application would require the latency to be minimized as opposed to a video streaming application which would require high bandwidth. Some large data processing applications might require to take into account the compute load at the destination servers along with network metrics. This idea is exemplified in [7] which motivates the advantages of network virtualization. A virtual network can in future allow for multiple service providers to offer their applications with customized requirements over a single physical substrate. Thus, being able to consider application specific requirements along with network parameters allows for improvement in the QoS and subsequently the user

experience.

In this report, we describe the design and implementation of a virtual network architecture over MobilityFirst. We discuss the features of the MobilityFirst protocol which are favorable for realizing this design with much ease. We use ASR (Application Specific Routing) as an illustration of the advantages of a virtual network deployment on MobilityFirst. A proof of concept validation and basic performance evaluation using classical networking tools, ping and iperf, ported to MobilityFirst are also presented. In this report we shall use scenarios with clients and a replicated service (at multiple sites) to illustrate the various design concepts and results.

Chapter 2

MobilityFirst

The Internet is facing a tremendous increase in mobile devices and applications which is on its way to replace the fixed-end point model that it was originally designed for. This has given rise to many clean-slate next generation architecture proposals to rethink the current IP-based design in order to cater to the shift towards mobile communications. MobilityFirst[2] is one such proposed Future Internet Architecture whose core idea is to support scalable mobility and wireless access. The need for such a clean-slate design to rethink the current internet architecture comes from the visibly growing number of mobile devices and the exponentially increasing traffic generated by them. The current internet architecture will not be efficient in supporting this imminent, ubiquitous and large scale wireless scenario.

The IP-based architecture is built on IP address assignments to each device based on the current location of that device. As a mobile device moves or changes its location frequently it needs to update its IP address at the same rate. This update in the location based address needs to communicated in the end-to-end communication system. MobilityFirst alleviates this overhead by using a Globally Unique Identifier(GUID) which ensures that the underlying network takes care of the location based forwarding of the message thus providing a seamless end-to-end communication without the need for frequent updating of the location-based information at the end points. At the narrow-waist of the MobilityFirst stack is the GUID (Globally Unique Identfier). The GUID is a unique and permanent identifier to reference any object in the MobilityFirst network. It can be used to identify routers, end hosts, content and even whole networks. This flexibility allows us to not only uniquely identify virtual nodes in the virtual network, but complete virtual networks themselves. We will, in this chapter, further describe the name resolution service called GNRS (Global Name Resolution Service) which makes it easy to lookup these

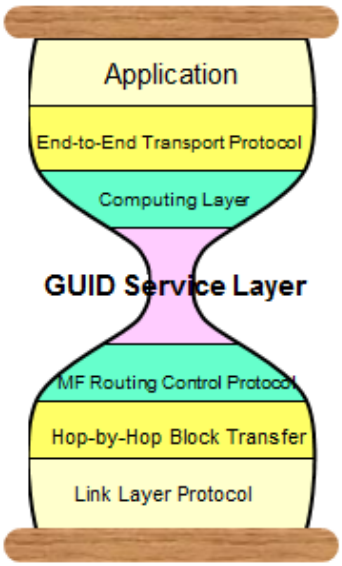

Figure 2.1: MobilityFirst stack with GUID at the narrow waist [8]

various indirections and other features of MobilityFirst.

2.1 MobilityFirst design goals

The main goals of MobilityFirst are to support high mobility and enhance trustworthiness. The following are some of the design points integral to MobilityFirst which are not well supported by the current IP architecture.

- *Separation of Names and Addresses:* MobilityFirst assigns to each object in the network a GUID which is a public key assigned by the Name Certification Service. These are long lasting network level names for any network object. The GUID is mapped to a set of network addresses which indicate the point of attachments of that object to the network.

- *Seamless host and network mobility:* Any host/ interface in the IP architecture is recognized by an IP address which identifies its network location. This is tedious when trying to support a highly mobile environment where the hosts are constantly moving and as a result the network location keeps changing. MobilityFirst defines 'flat' GUIDs(Globally Unique Identifiers) for each network

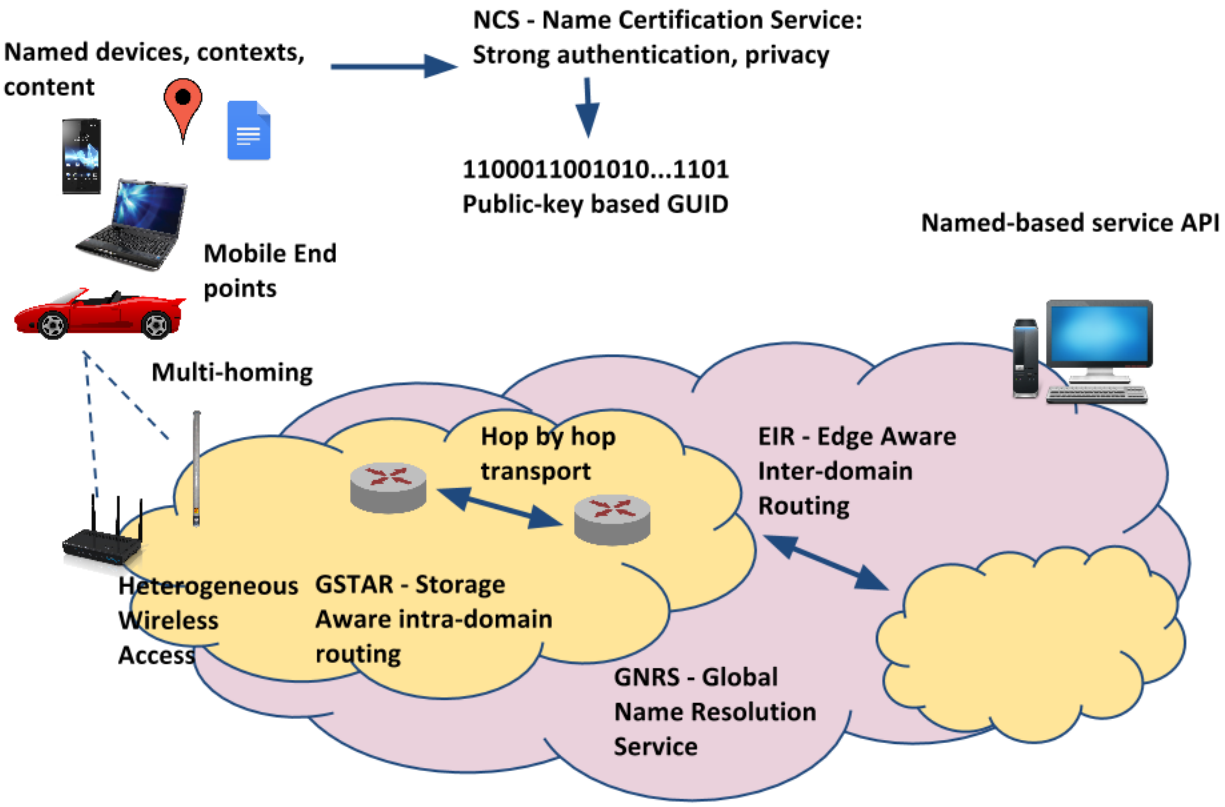

Figure 2.2: MobilityFirst High Level Network Architecture [9]

object. Packets are exchanged between sources and destinations addressed by the GUID while the underlying network handles routing the intended packets to the network location.

- *Name-based Network Service API:* The service API in MobilityFirst uses the unique names of the end point network objects instead of their network addresses or interfaces [2]. This allows for abstract services [10] based on multi-homing, multicast, named content, anycast etc. In MobilityFirst, since any network object can be named and identified by a GUID, context can be captured by assigning a GUID to a group within a certain location, time or to a service. We take advantage of the Service Anycast feature to identify replicated services in our use-case for Application Specific Routing.

- *Hybrid Name/ Address Based Routing:* The GUID and the NA (Network Address) mappings are carried in the PDU header to have a hybrid approach. The use of NAs provides a ”fast path” and the GUID resolution provides late binding. Late binding refers to the option that routers can query the GNRS to bind addresses to a certain GUID during forwarding. The **GNRS** (Global Name Resolution Service), a distributed service, is one of the key components that provides dynamic name to address resolution. For our work, we take advantage of the flexibility of what can be stored in the GNRS. We keep track of the Virtual network GUID (identifier for an entire virtual network) mapped to its member virtual nodes (router/ end hosts) which is indicative of the topology. We also maintain mappings between the virtual node GUID and the true GUID of the physical node it is a logical abstraction of.

- *Storage-Aware Intra-domain:* Routing in a MobilityFirst network is acheived by GSTAR (Generalized Storage Aware Routing) [11]. The core principles that form this component of MF are the separation of name and addresses, late binding of routable addresses, in-network storage and a conditional routing decision space. Routers have the option of temporarily storing data as *network-layer routing decision* which implies hop-by-hop transport.

- *Hop-by-Hop Transport:* MobilityFirst uses a hop-by-hop mechanism for transport. The entire file is received and stored at each node before sending to the next hop. In a highly mobile environment

where the location of the client may keep changing within the duration of the communication, we either face disconnections or have to keep setting up new TCP connections for each new IP address that the client acquires. With the advantage of GUIDs and the GNRS this need to keep track of the location at the end hosts is avoided. A continuous end to end connection need not be maintained since messages are communicated in the form of chunks between consecutive hops.

Other features include multi-homing, edge-aware inter-domain routing, in-network computing services.

Chapter 3

Virtualization

Virtualization refers to creating a virtual instance of a resource in order to support multiple instances on the same physical resource. Virtualization has two major approaches, one is to partition the available resources to allow multiple participants over the same physical substrate while the other is to aggregate a set of physical resources so as to give the participant a greater amount of resources to use than there is physically available on a single substrate. Three prominent areas where virtualization has been applied are :

- Server Virtualization: This involves partitioning the server resources to host multiple server instances on it. It can be achieved at different levels i.e. full virtualization, para-virtualization and OS-level virtualization [12].

- Storage Virtualization: The idea is to provide a layer of abstraction above the underlying physical storage resources. To a server, the storage virtualization layer provides a view different from what the true physical structure is. With virtualization these physical storage resources can be combined or partitioned [13].

- Network Virtualization: Network virtualization is the concept of running multiple logical networks atop (parallel) a common physical substrate. In our project we explore this area and motivate the advantage of a native virtual network with MobilityFirst to support the concept of Application specific routing(Section 3.1.3).

3.1 Network Virtualization

3.1.1 Background and Related Work

Server and storage virtualization have been given a lot of attention so as to improve the efficiency and ease of using these available resources by aggregating or partitioning them depending on the requirements. This idea was extended to networks so as to leverage the same advantages by partitioning or aggregating network resources. Partitioning a network allows for the support of multiple users in isolation, while aggregation of the resources allows for applications requiring increased bandwidth, memory, etc. Network virtualization means creating a logical view of networking resources such as routers, switches, links, etc. The physical devices are solely responsible for forwarding. while the intelligence resides in the virtual network where routing decisions and algorithms are known. The most commonly seen forms of virtual networking are VLANs (Virtual Local Area Networks), VPNs (Virtual Private Networks), active/ programmable networks and overlay networks. VLANs involve the partitioning or aggregation of physical LANs to have multiple logical LANs or a resource-rich single virtual LAN respectively. Our take-away from VLAN technology is to isolate virtual networks from each other while being hosted on the same physical network. A VPN involves connecting two end points via a tunnel over the existing network in order to provide a secure communication. Active networks allow for great flexibility in network customization. Programmability in active networks have a range of levels. On the one end we have active routers with the ability to execute any code, while on the other, routers have a set of existing functions to use. Overlay networks are primarily seen in the application layer. In our work we propose a virtual network implemented on the routing layer of MobilityFirst. Our design like many other in the literature [14] borrows ideas from VLANs and VPNs for their tunneling/ encapsulation mechanisms. Active/ programmable networks provide inspiration in their flexibility to support a variety of applications.

We are interested in supporting multiple virtual networks on a physical substrate running MobilityFirst. These various logical networks are expected to be isolated from each other and by taking the advantages of MobilityFirst features (Section 2) such as the GNRS, Name-based Service and Service Anycast, we

wish to build a case for the concept of Application Specific Routing (Section 3.1.3).

The network virtualization problem has been extensively studied and put down in literature. *Software defined networking*(SDN) is one such platform which supports network virtualization [15]. The core idea of SDN is to decouple the control and data planes so as to have programmable network control. In a regular switch the decision to route a packet is built into the switch's firmware. With SDN, the rules for forwarding are specified by the controller, which is the heart of the SDN architecture. The controller has a complete view of the network and is a single point of configuration from where the entire network can be programmed [16]. An important point to bear in mind is that virtualization is one of the many applications that can be supported by SDN and SDN is one of the platforms that supports network virtualization.

We follow the general guidelines defined in *CABO - Concurrent Architectures Better than One* [17], where a full virtualization architecture is presented emphasizing the separation of infrastructure providers from service providers. The main aspects useful for our design considerations are: support for tunneling and encapsulation, give service providers direct control over protocols and services that run on virtual nodes, interface for service providers to make requests, bootstrapping capabilities, admission control and finally network embedding.

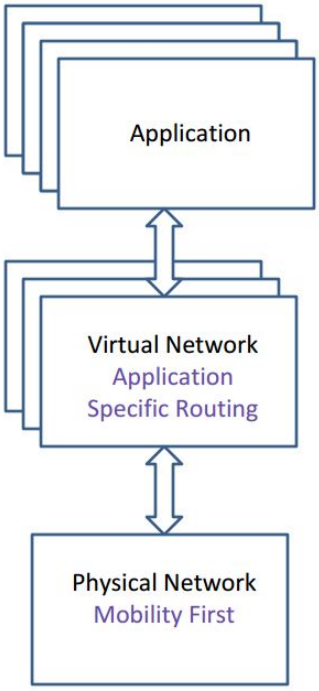

Figure 3.1: Multiple virtual networks(for multiple applications) can be deployed atop the physical substrate running MobilityFirst

Figure 3.1 shows the virtual network layer introduced above the physical substrate. This high level diagram implies that multiple virtual networks can be brought up on a single physical substrate and each of these virtual networks can cater to a single or multiple applications.

3.1.2 Virtual Network over MobilityFirst

Some of the key aspects that need to be addressed while designing a virtual network architecture are [14]: *bootstrapping*, *interfacing* between the service and infrastructure providers, *resource allocation*, *admission control*, *resource scheduling*, *naming and addressing*, *dynamism/ mobility* and *heterogeneity and isolation*. In our design, elaborated in Section 4, we propose a central controller which communicates to the infrastructure providers the requirements of a new virtual network, i.e. the topology, identity information. It is involved in the bootstrapping phase of each new virtual network. We propose an infrastructure provider interface which coordinates the various application requirements with the central

coordinator and accordingly ensures resource allocation. In our current implementation these components have not been realized yet, but they are part of the scheme to accommodate a real world scenario with multiple service and infrastructure providers.

Advantages of native implementation over MF

MobilityFirst's features inherently address some of the above mentioned aspects. Every object in the MobilityFirst network can be recognized by a GUID (Globally Unique Identifier). We assign to each virtual network a GUID and this helps to separate the routing tables (within a MF router) of the various virtual networks from each other. Additionally, with the help of the GNRS we can map to each Virtual Network GUID, a list of the member virtual nodes and the mappings of the virtual GUIDs to the true GUIDs for these members. Resource scheduling is required in virtualized environments where multiple logical instances run atop the physical infrastructure. In our native implementation of the virtual modules within the routing framework, scheduling is a matter of merely looking up the Virtual Network GUID to obtain a reference to the appropriate virtual routing table. Supporting heterogeneity of naming/ addressing in a multiple virtual network scenario is a difficult problem with the current IP architecture. Users should also be able to simultaneously connect to multiple virtual networks using different technologies (referred to as uber-homing [18]) in the case that various applications have a dedicated virtual network. MobilityFirst's multihoming feature is a counterpart to this concept. These cases require a means to recognize the end user globally as well as to separately reference the end user's connection to each virtual network. With the help of the single permanent name (GUID), MobilityFirst(MF) takes care of this problem as part of its design. [18] proposes a separation of identities of the physical and logical locations of a user, which in the case of MobilityFirst is achieved by simply assigning a GUID to the physical node and one to each of its virtual instances.

Some of our ideas such as a central controller to initiate the virtual network may be seen as similar to some design ideals of Software Defined Networking(SDN), but an important distinguishing aspect is that SDN requires redefining the network structure to have separate data and control planes while our

virtual network protocol can be natively supported by the existing design of MobilityFirst.

3.1.3 Application Specific Routing

Application specific routing is a mechanism wherein the routing layer metric is used in combination with application specific information to take routing decisions which could allow for more flexibility in terms of policies and could improve QoS. The application specifies an algorithm or method to combine certain metrics. A metric generated by the application end is called the node metric. For example. the CPU utilization factor at a server may serve as a node metric. A link metric identifies the state of the network as seen between routers. It could be an indicator of properties such as latency or bandwidth. The link metric would essentially reflect the routing fabric. A combination of the node metric and link metric is referred to as the application specific metric.

To illustrate this concept with an example, let us consider a replicated cloud service. In a regular cloud service scenario when a client requests a service it is directed to the cloud site which is closest to it in terms of shortest path. In a situation where this specific site is overloaded the client's request may either be dropped or redirected another cloud site slightly further away. With application specific routing we could avoid the drop of the request or redirection by taking a more informed routing decision early on by combining the routing distance along with the compute load at the cloud site. This way we can direct the client's request to a site who is slightly further away but will definitely be able to process incoming requests.

3.1.4 ASR Metric example

The application specific (node) metric is pushed into the network by the end application via the edge routers in the form of ASPs(Application Specific Packets). These ASPs are flooded to all other routers in the virtual network to populate the virtual routing table. This information is used as part of a combined metric to choose the appropriate destination for a service anycase request. The virtual nodes in the network, additionally, exchange among each other VNSPs(Virtual Network State Packets) containing

information about the node metric and the virtual link metric which are required to build the virtual routing table. The virtual link metric essentially reflects the routing layer metric. The routing information is retrieved from the routing tables of the physical router. In our experiments the routing metric is retrieved from the GSTAR routing table.

One of the design aspects to be addressed here is translation of this combination of parameters into a single metric for the routing on the virtual layer. One method to calculate this combined metric would be to use a weighted average of the normalized metrics assuming they have the same trend. For example, we use waiting time at a cloud site as the node metric and the estimated file transfer time as the link metric. The estimated file transfer time is the SETT value obtained from the GSTAR routing table scaled to the file size being transfered. We can apply an algorithm which adds these two time components to compute the ASR metric and chooses the destination having a lower metric value.The table below and Figure 3.2 are an example of how this can be achieved.

Destination	File Transfer Time (FTT)	Waiting Time	ASR = FTT + Waiting Time (in sec)
D1	20	63	20 + 63 = 83
D2	72	39	72 + 39 = 111

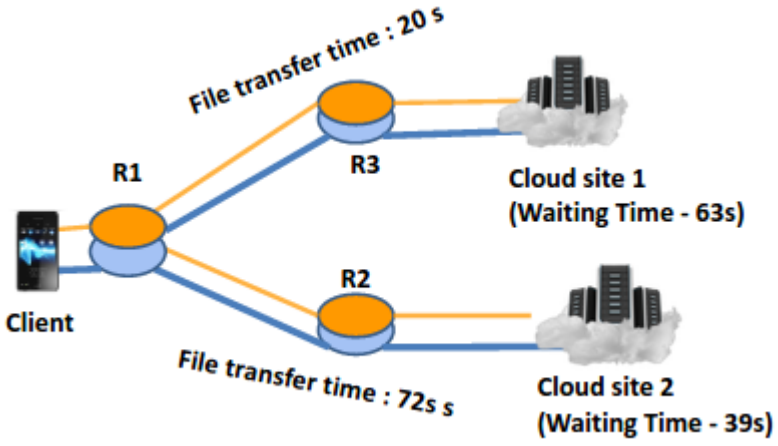

Figure 3.2: Weighted average to compute ASR metric

Another mechanism to make routing decisions is the use of a decision graph in which we segregate

the space into regions where certain routing decisions are made, Figure 3.3.

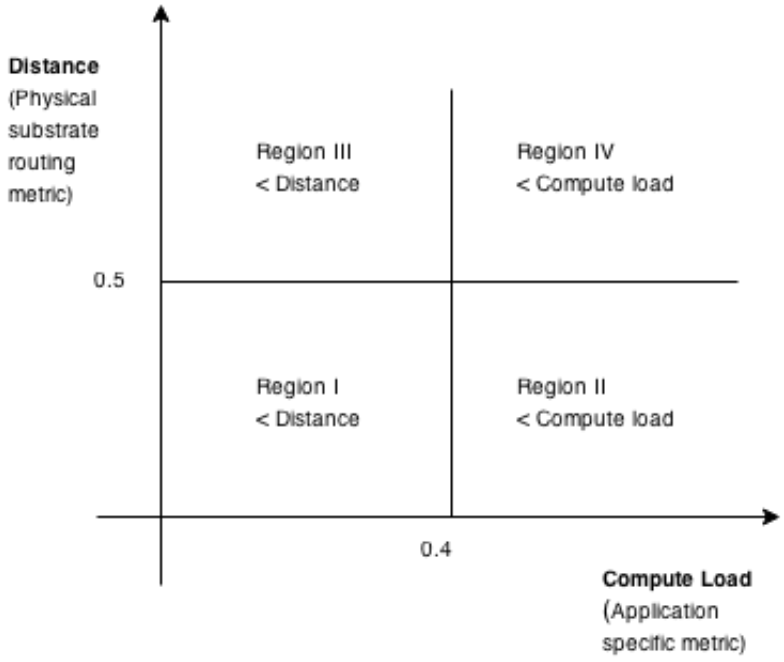

Figure 3.3: Decision Graph method to compute ASR metric

Chapter 4

Design of Virtual Network support over MobilityFirst

MobilityFirst is well suited to building a Virtual Network which supports Application Specific Routing. Since every object in the MobilityFirst network has a GUID, an application, service or even a whole virtual network can be identified by a GUID. We leverage the advantages of the name-based architecture, the GNRS (Global Name Resolution Service), support for Anycast delivery service and the ability to introduce extended computing logic into the routing fabric. The MobilityFirst name-based architecture allows us to assign to each Virtual network an ID that uniquely identifies it. Each router in our design can accommodate multiple virtual router instances, and each such instance is assigned a GUID. This helps to isolate the multiple virtual networks on the same physical network. The GNRS keeps track of various mappings, the Virtual Network's GUID to its member virtual router GUIDs and each virtual router GUID to its true GUID. In a MobilityFirst network since every component can be identified by a GUID, we have the facility of assigning to an application or service a unique GUID as well. This combined with the Anycast feature makes it conducive to supporting the idea of Application Specific Routing.

Consider the example with multiple cloud sites running the same application. A client can request a service identified by the service GUID and the request would be routed to the cloud site which is the best option based on the routing as well as the application specific metric. With service-anycast, the client solely needs to specify the GUID of the service and will be routed to the best (in terms of a combined metric) destination site.

While defining the virtual network for a certain application we assign a GUID for a virtual network and a distinct GUID for each virtual node in that network. In order to keep track of these names and

correspondences we take advantage of the GNRS. In the GNRS we can maintain the mappings between the virtual network GUID and its component virtual nodes. We can also keep track of the virtual GUID to true GUID mapping of each node.

Since MobilityFirst implements a Hop-by-Hop transport mechanism, if at any point during the routing of a request the application or the routing state changes, it could help take a more informed routing decision even after the request has made it downstream.

4.1 Design Overview

The goal is to design a virtual network architecture that allows for custom topology and routing among the resources while ensuring isolation. With this architecture we can realize application specific routing (ASR) to be deployed in a virtual network atop the physical substrate running MobilityFirst. This section lists the design details of this architecture in terms of the parameters, devices and technologies required.

4.1.1 Components

- *Central coordinator*: The network requires a central coordinator or interface to which the user (service provider) can specify the topology in terms of the desired number of nodes and link specifications. We consider that the any state changes in the physical substrate such as link/ node failure, etc. will be communicated to the virtual network so that appropriate further routing decisions can be made. The central controller sends the topology file to each of the nodes in the virtual network.

- *Resource allocation*: The virtual resources may be realized in various ways. In some cases the substrate resources may be virtual to begin with, resulting in nested virtualization. For our purposes we consider a single layer of virtualization but we do support parallel virtual networks on the physical substrate. Usually, a single substrate resource is partitioned into multiple virtual network resources. A node on the physical substrate would host a number of virtual machines and a physical link may be able to support multiple virtual links. In some other cases, physical substrate resources

may be combined to form a virtual resource. In our experiments we consider the former.

- *Infrastructure provider interface/ API*: This interface, at the infrastructure provider's end, talks to the central coordinator about available resources for the given request. It would load the appropriate images and applications on to the virtual node as directed by the central coordinator. For our experiments we excluded this device since we were not dealing with multiple infrastructure providers.

- *Virtual GUIDs*: Every virtual network is identified by a Virtual Network GUID (VN_GUID) and each virtual node on a device has a Virtual Node/ Machine GUID (VM_GUID). In other words, if a certain device supports multiple virtual nodes belonging to different virtual networks, each of those virtual nodes will be assigned a unique identifier.

- *GNRS*: The GNRS is a distributed name resolution service which maintains mappings between the GUIDs and the list of network addresses. The GNRS also has the flexibility to store any other relevant mappings such a multicast group GUID to its list of component GUIDs. We take this advantage of having various kinds of mappings in the GNRS into our design of the virtual network. The following correspondences are stored in the GNRS to cater to the virtual network:

  - GUID of the virtual network - Virtual GUIDs of the member nodes
  - Virtual GUID of a virtual node - true GUID of the node
  - Service GUID - set of virtual GUIDs of destination nodes that the replicated service is hosted on

- *GSTAR Routing*: The physical substrate on top of which the virtual network is built runs the MobilityFirst protocol stack i.e. uses the GSTAR routing protocol [11] to move packets across the network. The GSTAR routing tables are populated based on the link state messages that are exchanged among the router nodes. This routing table information is made available to the virtual layer in order to compute a resultant metric along with application specific routing information exchanged at the virtual layer.

- *Application Specific Routing*: The routing on the virtual layer is commanded by the application specific metric, explained in 3.1.3.

- *Packet forwarding and Encapsulation*: Once the virtual layer takes a decision on who the destination node is, it details the packet header with virtual GUIDs of the source and destination nodes. The virtual routing layer informs the routing layer of the virtual GUID of the next hop on the virtual network. This packet is encapsulated to contain as the destination address, the true GUID of the node which is the next hop in the virtual path. The physical substrate routes this packet to the node of the next virtual hop by the GSTAR protocol.

  In MobilityFirst the Packet header has a provision for an Extension header. Utilizing the extension header would be an equivalence to encapsulating the virtual packet with the routing header. The stripping of the header when using encapsulation is not a trivial implementation due to which using the extension header was given due consideration.

4.2 Design Details

We shall describe some of the important design aspects such as the routing tables, for the physical and virtual layers and the packet headers using the following topology as an example in Figure 4.1. The blue nodes depict the physical routers and the orange nodes are the virtual instances forming a single virutal network. A client request follows the virtual path in orange which implies a physical transfer via the blue path.

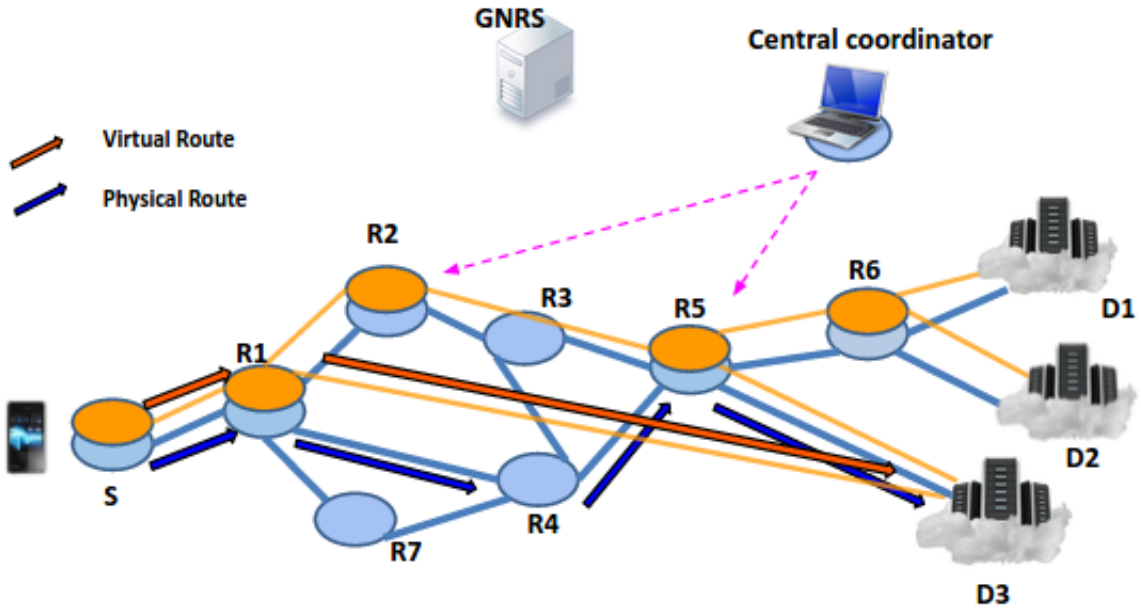

Figure 4.1: Topology

4.2.1 Routing and Virtual Routing tables

The forwarding tables of the GSTAR routing is available to the virtual network to use for further computation of the virtual layer's forwarding table. The Virtual Forwarding tables calculate the best next hop for a given destination with the help of the virtual network state packets that are exchanged among the nodes. These control messages are of two types: the first generated by the routers in the network which carries the node and link metric information about its immediate neighbors, and the second is pushed into the network by the application end hosts.

GUID Dest	GSTAR Metric: SETT/ LETT	GUID Next
S	a	R1
R2	b	R1
R3	c	R1
...	...	...
D1	x	R1
D2	y	R1
D3	z	R1

VM_GUID Dest	Virtual Metrics		VM_GUID Next
	Node Metric e.g. Load	Vlink cost e.g. path length	
VM_S	*	a'	VM_R1
...	*		...
VM_D1	X	x'	VM_R2
VM_D2	Y	y'	VM_R2
VM_D3	Z	z'	VM_D3

Figure 4.2: High level design of Forwarding and Virtual Forwarding Tables

4.2.2 Packet Headers for Encapsulation

We use the extension header space (88 bytes) in the MobilityFirst routing header to carry virtual header information so as to route packets on the virtual layer. This structure is similar to encapsulating a virtual packet with the routing header. Routing at the virtual layer establishes the next virtual hop for a given virtual destination. For the exchange between two virtual hops, the packet routing header is assigned the true GUID of the next virtual hop as its destination. Section 4.2.4 shows an example of the message flow.

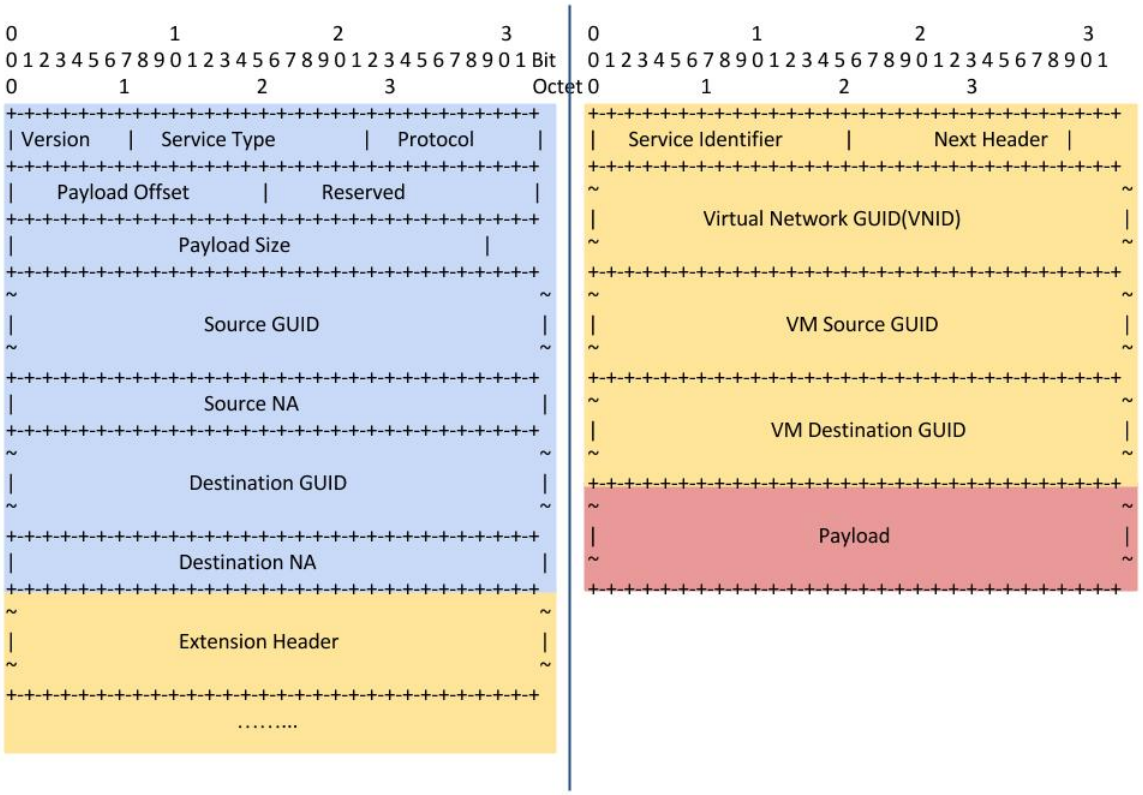

Figure 4.3: Packet encapsulation Headers

The control information carried in the packet header are shown in Figure 4.3. The routing header contributes to 60 bytes and the virtual extension header is 64 bytes.

4.2.3 Control Information

In order to build a GSTAR (Generalized Storage Aware Routing) routing table in MobilityFirst, routers exchange their link state information in the form of LSPs (Link state packets). Similarly to build the virtual routing tables, the virtual routers exchange VNSPs (Virtual Network State Packets) and ASPs (Application State Packets) are introduced into the network by the end host applications. These data structures are detailed in Section 5.2.

4.2.4 Example of message flow for encapsulation

Here we give an example of how the encapsulation works for a message expected to go from Source S to Destination D3 on the virtual network. The sequence of events involved are depicted by Figure **??**.

- Figure 4.4: Router S connected to the client requires to route the message towards Virtual destination VM_D3. It looks up the virtual routing table for VM_D3 and retrieves the virtual next hop VM_R1. This virtual message is encapsulated with the physical substrate's routing header such that the destination is R1.

- Figure 4.5: When the packet reaches next hop R1, it looks at the GSTAR routing header to find that the destination is itself. In this case it pushes the packet to the upper protocol - virtual processing elements for virtual router VM_R1.

- Figure 4.6: VM_R1 looks up the its virtual routing table to retrieve the next virtual hop for the destination VM_D3. The virtual next hop is VM_D3 itself and the routing layer encapsulates this packet with D3 as the destination.

- Figure 4.7: The packet is forwarded via GSTAR routing to the virtual next hop VM_D3 which is the destination (D3) for the GSTAR routing layer.

- Figure 4.8: This packet intended for VM_D3 upon reaching R5 does not get pushed to the upper protocol (virtual) processing elements since it is not the intended destination for the GSTAR routing layer and is thus, forwarded as a regular packet towards D3.

- Figure 4.9: D3 receives the message intended for itself and pushes it to the virtual router VM_D3 which is the destination for the virtual header.

S

GUID Dest	Metric	GUID Next hop
R1	a	R1
R2	b	R1
R3	c	R1
R4	d	R1
R5	e	R1
R6	f	R1
R7	g	R1
D1	h	R1
D2	i	R1
D3	j	R1

VM_GUID Dest	Metric	VM_GUID Next hop
VM_R1	A	VM_R1
VM_R2	B	VM_R1
VM_R5	C	VM_R1
VM_R6	D	VM_R1
VM_D1	E	VM_R1
VM_D2	F	VM_R1
VM_D3	G	VM_R1

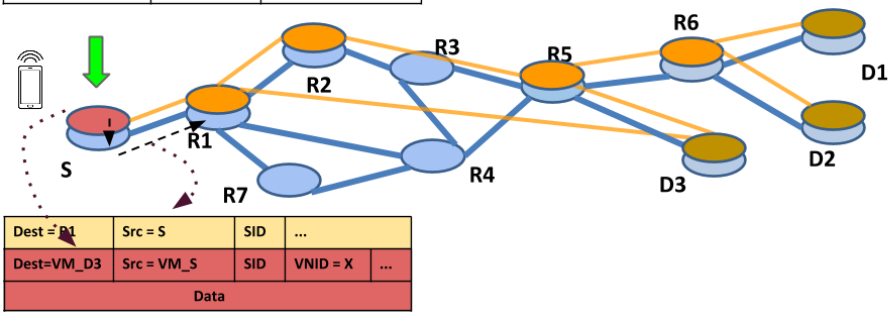

Figure 4.4: Message starts from Source S

R1

GUID Dest	Metric	GUID Next hop
S	a	S
R2	b	R2
R3	c	R2
R4	d	R4
R5	e	R4
R6	f	R4
R7	g	R7
D1	h	R4
D2	i	R4
D3	j	R7

VM_GUID Dest	Metric	VM_GUID Next hop
VM_S	A	VM_S
VM_R2	B	VM_R2
VM_R5	C	VM_R2
VM_R6	D	VM_R2
VM_D1	E	VM_R2
VM_D2	F	VM_R2
VM_D3	G	VM_D3

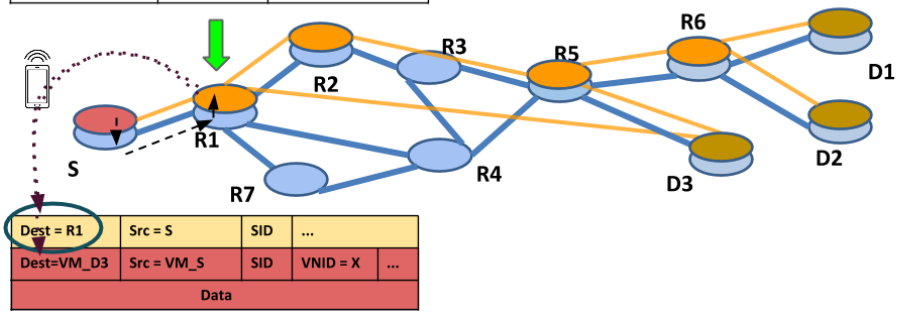

Figure 4.5: Message at first Virtual hop R1

GUID Dest	Metric	GUID Next hop
S	a	S
R2	b	R2
R3	c	R2
R4	d	R4
R5	e	R4
R6	f	R4
R7	g	R7
D1	h	R4
D2	i	R4
D3	j	R7

R1

VM_GUID Dest	Metric	VM_GUID Next hop
VM_S	A	VM_S
VM_R2	B	VM_R2
VM_R5	C	VM_R2
VM_R6	D	VM_R2
VM_D1	E	VM_R2
VM_D2	F	VM_R2
VM_D3	G	VM_D3

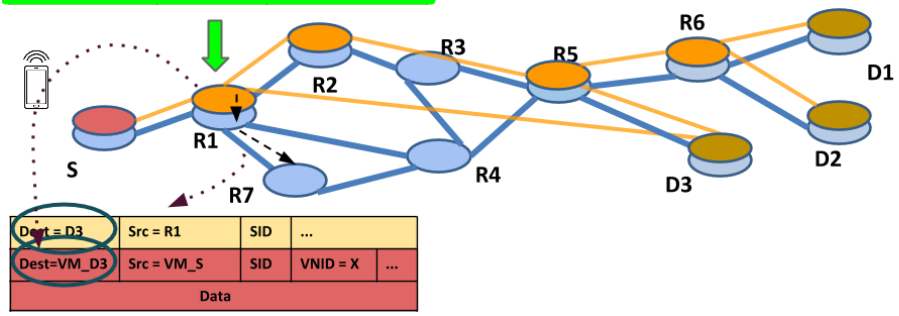

Figure 4.6: Routing header updated to carry new destination GUID

GUID Dest	Metric	GUID Next hop
S	a	R7
R1	b	R7
R2	c	R3
R3	d	R3
R5	e	R5
R6	f	R5
R7	g	R7
D1	h	R5
D2	i	R5
D3	j	R5

R4

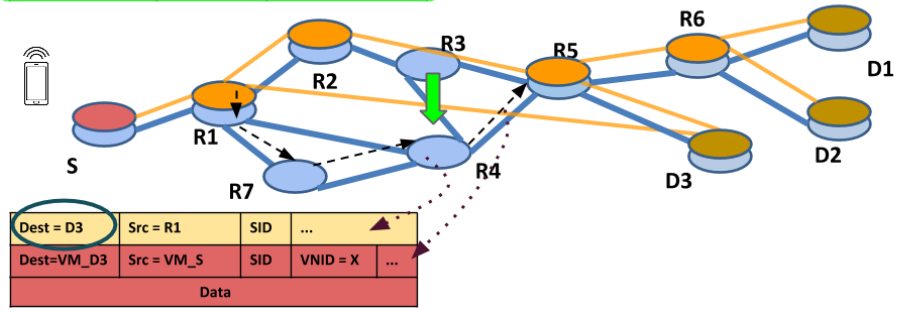

Figure 4.7: En route to next virtual hop D3

GUID Dest	Metric	GUID Next hop
S	a	R4
R1	b	R4
R2	c	R3
R3	d	R3
R4	e	R4
R6	f	R6
R7	g	R4
D1	h	R6
D2	i	R6
D3	j	D3

R5

VM_GUID Dest	Metric	VM_GUID Next hop
VM_S	A	VM_R2
VM_R1	B	VM_R2
VM_R2	C	VM_R2
VM_R6	D	VM_R6
VM_D1	E	VM_R6
VM_D2	F	VM_R6
VM_D3	G	VM_D3

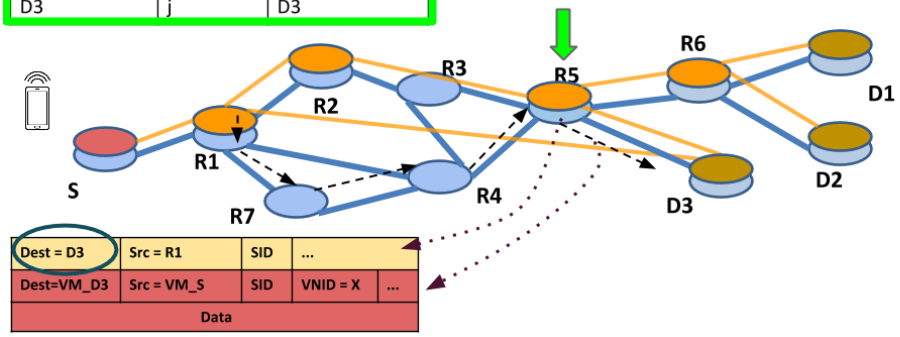

Figure 4.8: R5, although a virtual router, is not the routing destination; message routed to D3 via GSTAR

GUID Dest	Metric	GUID Next hop
S	a	R5
R1	b	R5
R2	c	R5
R3	d	R5
R4	e	R5
R5	f	R5
R6	g	R5
R7	h	R5
D1	i	R5
D2	j	R5

D3

VM_GUID Dest	Metric	VM_GUID Next hop
VM_S	A	VM_R1
VM_R1	B	VM_R1
VM_R2	C	VM_R5
VM_R5	D	VM_R5
VM_R6	E	VM_R5
VM_D1	F	VM_R5
VM_D2	G	VM_R5

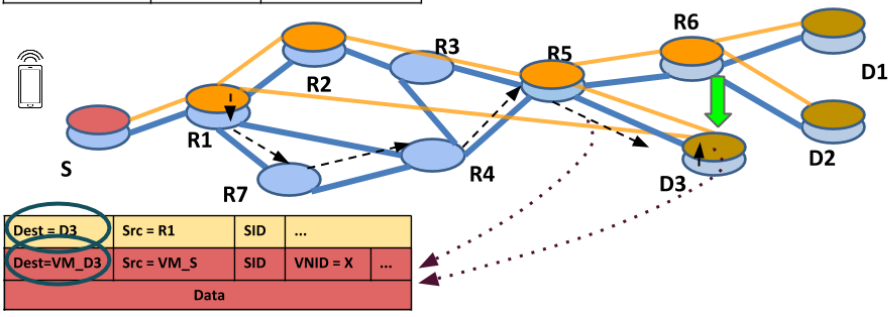

Figure 4.9: Message reaches intended virtual destination

Chapter 5

Proof of Concept Implementation

Our implementation is incorporated as native elements within the MobilityFirst Click routers. The design concepts explained in chapter 4 have been in implemented in order to be integrated into the MobilityFirst FIA prototype.

The following are the stages of implementation that have been envisioned as the first step:

- Virtual router to choose the next hop based on ASR metric. The GSTAR routing protocol is used between hops
- Support for Multiple Virtual Networks
- Integrating with a Global Name Resolution Service

5.1 Virtual Router

The virtual router is implemented using click elements integrated with the MobilityFirst FIA's prototype router. We start of with the assumption that a central coordinator (CC) wants to create a virtual network on top of a set of physical resources already available/ assigned to it. The CC is aware of the GUIDs of these routers/ nodes.

- The first step is to provide the physical substrate topology and virtual network topology. At this stage we emulate the GNRS functionality by providing the mappings between the true and virtual GUIDs and the mappings for the service GUID to its component node GUIDs.
- Virtual control messages fall under two categories, the first are the messages exchanged among the routers and the second is from the application which conveys to the network the node metric. The

former control message primarily comprises the router's virtual link cost to it's neighbors and it's own node metric.

- Although the virtual control messages are exchanged in the virtual network to update the forward logic, these messages are communicated data packets (i.e. MobilityFirst chunks) in the true network.

- The virtual forwarding logic makes use of the Virtual Routing table to decide who the destination should be. The implementation also supports resolving a service GUID to the right destination node based on an algorithm implemented for Application Specific Routing.

Click Elements

Figure 5.1 depicts the pipeline of elements within a MobilityFirst Click software router. In addition to the elements in the pipeline the following objects are created which are required by the virtual routing scheme:

Virtual Neighbor Table:

Two maps comprise the Virtual Neighbor table, the first keeps track for each virtual neighbor relevant information such as the node metric, virtual link cost to it, etc. while the second is a map which captures the virtual network link state messages(virtualNSP) that are exchanged among the routers.

Virtual Routing Table:

This element maintains a set of Hashtables:

- Virtual Forwarding Table

  This comprises forwarding information i.e. for a given destination it maintains the costs to it (routing cost as well as application specific costs) and the next hop. This table is built using Djikstra's algorithm based on the least link cost.

- Virtual Service Map

  With the service anycast feature of MobilityFirst we can specify a GUID for a certain application

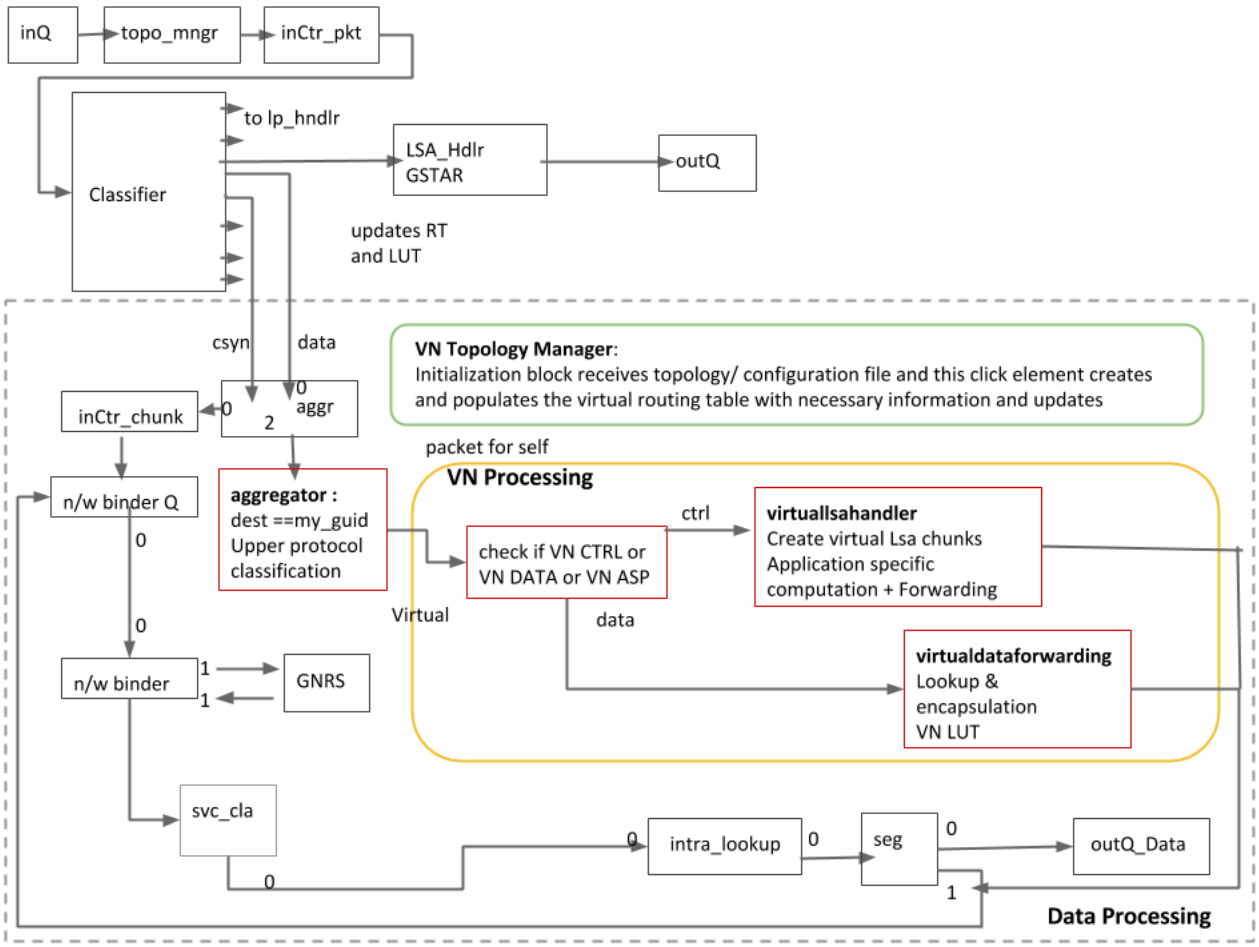

Figure 5.1: MobilityFirst Click Software Router with VN Processing

and this GUID would map to a set of node/ destination GUIDs. This mapping is held in the Virtual Service Map, another component which temporarily emulates a name resolution functionality.

- Virtual ASP Map

  This map captures the Application specific information that is pushed into the network by the application end points in a control message called ASP.

Virtual GUID Map:

This element emulates the GNRS functionality in the current stage of the implementation. It is a hash table which has the mappings between the virtual network GUIDs of the nodes in the network to their true GUIDs. This element would ideally be replaced with integration with the GNRS.

The elements implemented within the MobilityFirst Click router to process packets intended for the virtual network are the following:

- *Virtual LSA Handler:*

  This element handles the two types of virtual control messages that are exchanged in the virtual layer. The first is a virtual network state advertisement which is exchanged among routers. It is initiated by a router and carries information about the node and virtual link metric with respect to its neighbors. The second type of control message that propagates the network is one that is initiated by the end host/ application. In the case of a replicated web service/ data processing application, the application state advertisement would carry a metric which is equivalent to the compute load at the node running the application.

- *Virtual Data Forwarding:*

  Data packets that are to be routed through the network to a destination which may be a service or a specific destination are handled by this element. If it is intended for a Service GUID, this element uses the virtual forwarding table to figure out who the next virtual hop should be by estimating the best virtual destination for the combined ASR metric view that it currently has.

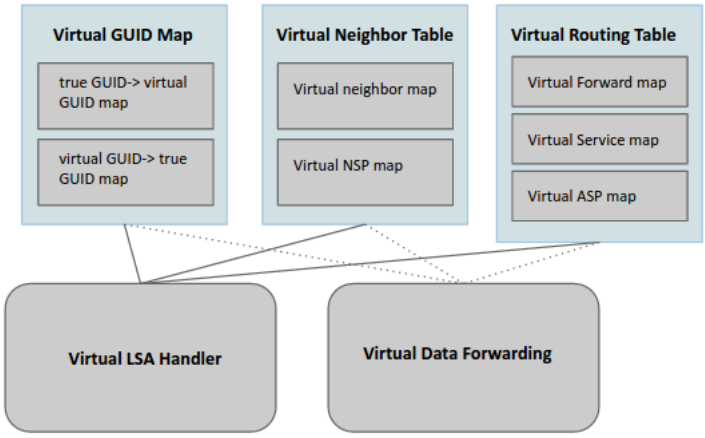

Figure 5.2: Single virtual network instance

5.2 Control Information

Similar to GSTAR's method of exchanging LSPs (Link State Packets), we have designed the virtual network routers to exchange state information as well. The control information exchanged are of two types:

- VNSP (Virtual Network State Packet):

  These control messages are similar to the LSPs(Link State Packets) exchanged as part of the GSTAR routing protocol. The VNSPs carry the virtual node metric at the router and the virtual link metrics to each of the router's virtual neighbors. This information is used to build virtual routing tables.

- ASP (Application Specific Packet):

  This control message is injected into the virtual network by the end application connected to the edge routers. It contains the application specific metric/ node metric at the end host. These ASPs are further forwarded to all other routers in the virtual network to populate the virtual routing table with entries which are used to resolve service anycast requests.

5.3 Multiple Virtual Networks

Although the implementation described in Section 5.1 currently supports a single virtual router instance on a MobilityFirst router, the same logic is applicable to a multiple virtual network scenario by extending the data structures to support it. Figure 5.3 shows how the single virtual network instance in Figure 5.2 can be extended to support multiple virtual instances.

Click Elements

In addition to the data structures described above in Section 5.2 we introduce a *Virtual Network Information* class and a *Virtual Topology Manager*. The Virtual Network Information class encapsulates all the required details about a single virtual network. The Virtual topology Manager maintains primarily a Hashtable of these various Virtual Network Information objects identified by the Virtual Network GUID for each VN. Our design for a native implementation of the virtual network, as opposed to a virtualized

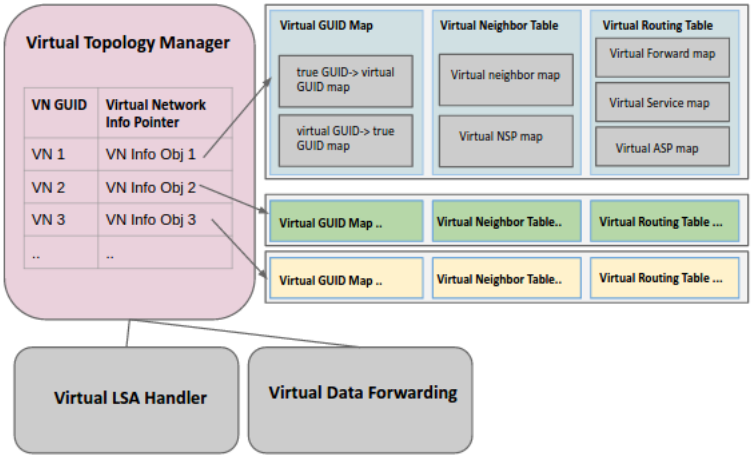

Figure 5.3: Multiple virtual network instances

environment, brings down the scheduling problem of switching among the multiple virtual instances on a single router to merely a table lookup by the proposed *virtual topology manager*.

Chapter 6

Evaluation

6.1 Proof of Concept Validation

A number of simple tests were conducted to ensure that the virtual network was indeed able to perform service anycast using the application specific metric. The following is an experiment that was setup to test the correctness of a simple threshold based algorithm to demonstrate Application Specific Routing(ASR).

Experiment Setup

The network comprises 12 routers of which 7 were part of a virtual network (depicted as the yellow nodes in 6.1). The edge routers denoted by virtual GUID 721 and virtual GUID 722 are connected to servers with GUIDs 21 and 22 respectively. These servers emulate a request generator to render a metric which is sent through the network as the Application Specific Metric within a control message called ASP. The virtual routers additionally exchange Virtual Network State messages (VNSP) carrying information about their neighbors. These control messages are used to build the virtual routing tables. The client with GUID 1 is connected to the virtual router with GUID 711 and periodically sends requests. The virtual network routes this request to the appropriate server by using the following algorithm:

Algorithm1

- If the minimum node metric is greater than the threshold, route to the server which is located the shortest number of hops away
- Else from among all the servers whose node metrics are lesser than the threshold choose the server which is located the shortest hops away as the destination.

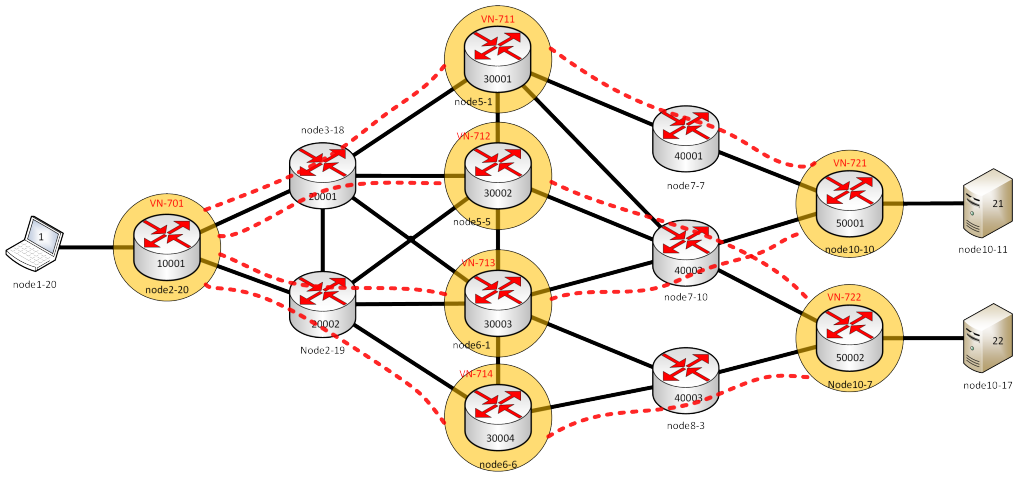

Figure 6.1: Experiment to test Application Specific Routing

The experiment was setup such that the two destination servers would inject into the virtual network their node metric which is a value between 0 and 1. The threshold used to choose the server was 0.5. The two destinations were programmed so as periodically and alternately push a value on either side of 0.5 into the network. Figure 6.2 depicts for each request, the values of the metrics at each destination as seen by the edge router and the choice made. 21 and 22 are the GUIDs of the destinations. The blue line corresponds to the metric values at server with GUID 21 and the red corresponds to the metric values at server with GUID 22.

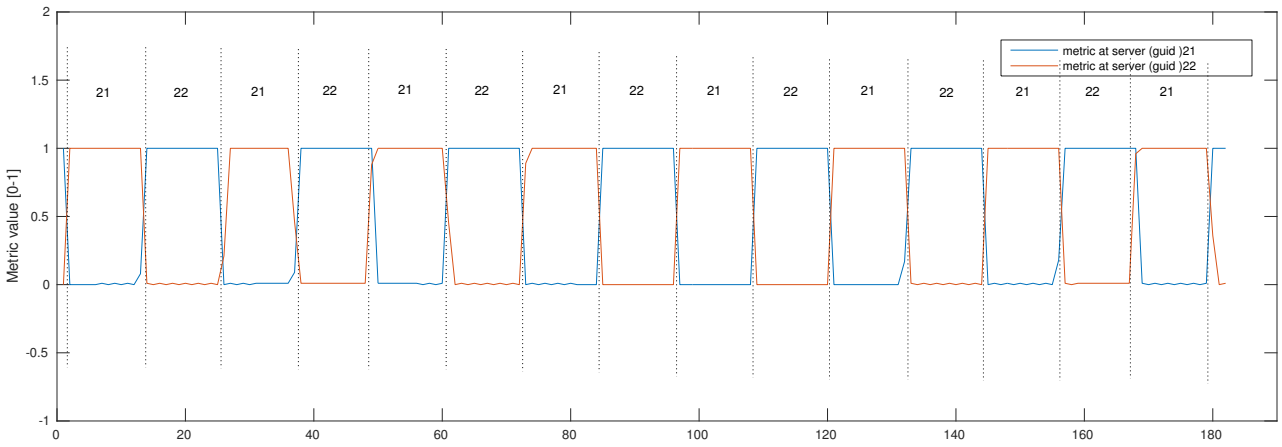

Figure 6.2: Application specific routing based choice

6.2 Microbenchmarks

The virtual network which runs over the MobilityFirst based physical substrate has its own routing algorithm in place based on the application specific metric. In order to compute the virtual routing table and other such resources a number of virtual control messages need to be exchanged among the routers of the network. Also, since the application specific metric is provided by the end host application, control messages which carry this information also need to be passed around. These control messages add to the overhead which would as a result affect the performance of the virtual network. The virtual network components are implemented as modules within the Click software router which implies additional processing for virtual packets that traverse the network.

In order to study these effects we designed and performed two experiments, the first was to measure the average round trip time taken by a ping message between a client and a server; and the second was an iperf experiment [19].

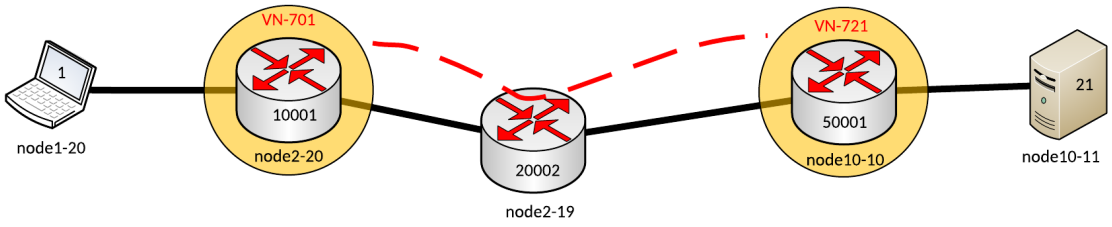

Figure 6.3: Experiment Setup for Microbenchmarks

These experiments were carried out on the Orbit testbed [20] at WINLAB. The individual nodes were part of the grid setup. For both these experiments we observed55results first without the virtual network and then with the virtual network implementation. This helps us compare and contrast how much of a difference the virtual network introduces.

6.2.1 Ping

”Ping” is a popular measure of network performance. It measures latency by calculating the average round trip time (RTT) between a source and destination. We measured the RTT for varying chunk sizes

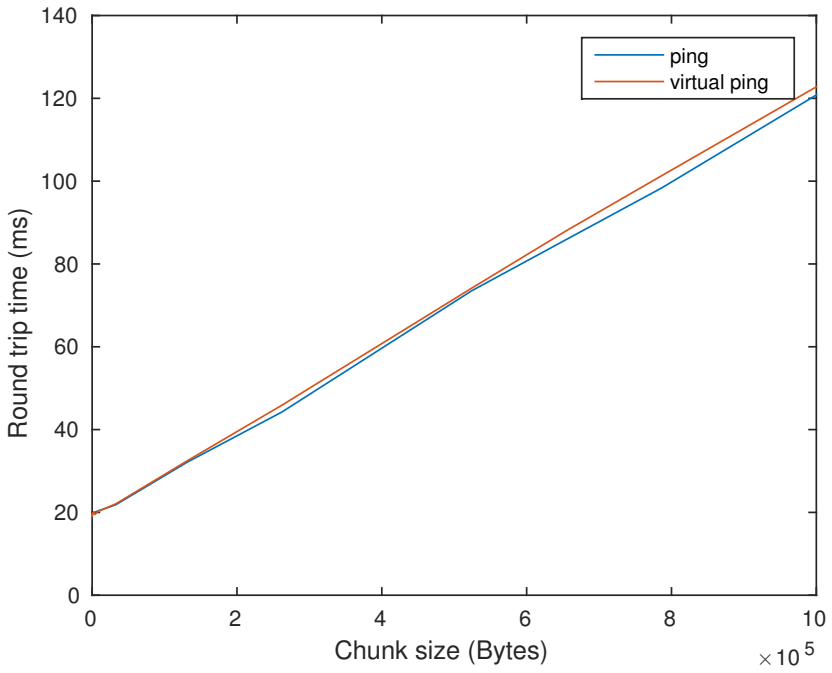

Figure 6.4: Ping characteristics for single Virtual network compared with MobilityFirst substrate

over the topology in Figure 6.3. We compared the results for the given topology with and without the virtual network components. Figure 6.4 shows the ping characteristics for both cases are very close indicating that the virtual network components which are part of the click implementation do not introduce a significant processing latency.

6.2.2 Iperf

Iperf is another classical networking testing tool which helps to measure the throughput of a network by creating data streams. We use the iperf tool ported to MobilityFirst which uses UDP-like data streams. For the topology in Figure 6.3 we compared the results for iperf with and without the virtual network and obtained the characteristics as seen in Figure 6.5.

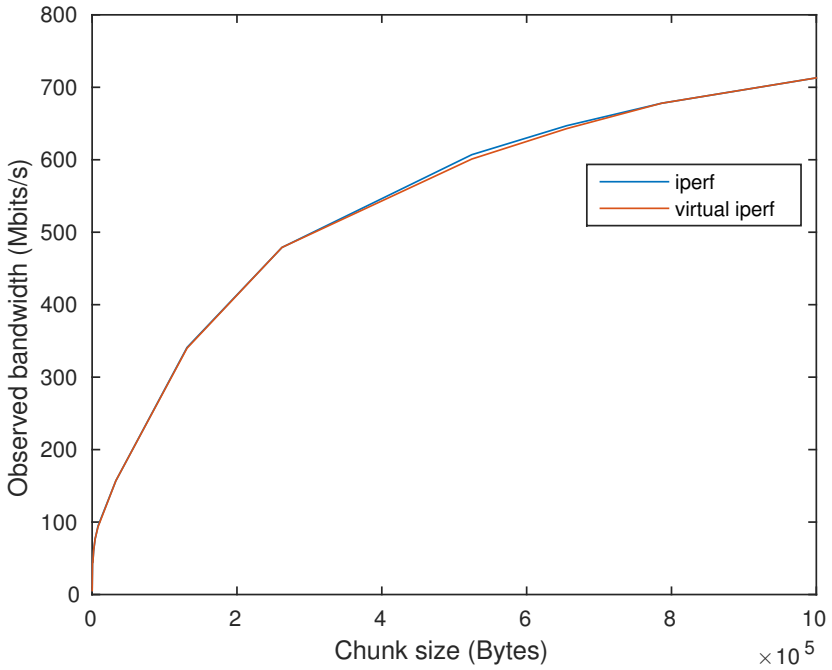

Figure 6.5: Iperf characteristics for single Virtual network compared with MobilityFirst substrate

6.2.3 Protocol and Control overhead

Protocol Overhead

The network or protocol overhead for the virtual network reflects the percentage contribution of control information in a data packet. In addition to the GSTAR routing header (60 bytes) information, the extension header of MF which is utilized to carry control information for the virtual network (64 bytes) contributes to protocol overhead. The MTU (Maximum Transmission Unit) size is restricted to that allowed by Ethernet which is 1500 bytes. Thus, the protocol overhead for our design works out to about 8.3%.

Control Overhead

Virtual routers exchange Virtual Network State Packets (VNSPs) carrying information about the link state with respect to their virtual neighbors. This adds to the control overhead within the network. For a topology with 20 nodes and 356 edges the overhead due to these exchanges would be about 156 KBps.

Chapter 7

Conclusion and Future Work

This work aims to motivate a virtual network for MobilityFirst by leveraging its various inherent design components such as name based object identification, the GNRS, ability to introduce extended computing logic into the routing fabric and service anycast. We have presented the design, implementation details and evaluation results for the current version. In our evaluation we have seen that the introduction of virtual elements into the software router shows little or no performance deterioration. We have demonstrated a use-case for this virtual network called application specific routing and have evaluated the implementation for correctness.

7.1 Future Work

7.1.1 Cloud Services Use-case Integration

In order to illustrate application specific routing over the proposed virtual network over MobilityFirst, we are working on integrating the current implementation of the VN with a replicated cloud service application. Two metrics form the basis of routing: edge performance metric and link performance metric. The edge performance metric is defined as the waiting time at a cluster which reflects the CPU utilization/ how heavily loaded that site is. The link performance metric is defined as the estimated file transfer time which reflects the bandwidth and latency of the link.

7.1.2 Design components

Some design aspects require to more in depth discussion. Bootstrapping the network involves introduction of the routing algorithm specific to the application and passing information to the member routers.

Complete functionality of the central coordinator and resource allocation are aspects which need more thought.

7.1.3 Multiple Virtual Network Support

An important next step is to complete the implementation of multiple virtual network support which is underway. Section 5.3 details the design.

References

[1] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, "A brief history of the internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.

[2] K. N. Dipankar Raychaudhuri and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, pp. 2 – 13, July 2012.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, no. 4, pp. 34–41, 2005.

[4] J. S. Turner and D. E. Taylor, "Diversifying the internet," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 2, pp. 6–pp, IEEE, 2005.

[5] http://teamarin.net/2014/08/13/transition-ipv6-taking-long/.

[6] S. Velrajan, "Application-aware routing in software-defined networks."

[7] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[8] http://mobilityfirst.orbit-lab.org/.

[9] http://mobilityfirst.winlab.rutgers.edu/.

[10] F. Bronzino, K. Nagaraja, I. Seskar, and D. Raychaudhuri, "Network service abstractions for a mobility-centric future internet architecture," in *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture*, pp. 5–10, ACM, 2013.

[11] G. B. Samuel C. Nelson and D. Raychaudhuri, "Gstar: Generalized storage-aware routing for mobilityfirst in the future mobile internet," *MobiArch '11 Proceedings of the sixth international workshop on MobiArch*, pp. 19–24, 2011.

[12] http://searchservervirtualization.techtarget.com/definition/virtualization.

[13] http://virtualizationreview.com/articles/2014/08/01/virtualized-vs-software-defined.aspx.

[14] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[15] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 20–27, 2013.

[16] http://www.infoworld.com/article/2606200/networking/111753-Software-defined-networking-explained.html.

[17] L. G. N. Feamster and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Computer Communication Review*, vol. 37, pp. 61–64, Jan. 2007.

[18] N. M. K. Chowdhury, F.-E. Zaheer, and R. Boutaba, "imark: An identity management framework for network virtualization environment," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*, pp. 335–342, IEEE, 2009.

[19] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 3–14, ACM, 2006.

[20] http://www.orbit-lab.org/.