Cut-Through Switching Options in a MobilityFirst Network with OpenFlow

Adrian Lara , Byrav Ramamurthy
University of Nebraska-Lincoln
{alara,byrav}@cse.unl.edu

Kiran Nagaraja, Aravind Krishnamoorthy , Dipankar Raychaudhuri
Rutgers University
{nkiran,aravind,ray}@winlab.rutgers.edu

***Abstract*—Mobile devices are expected to become the Internet's predominant technology. Current protocols such as TCP/IP were not originally designed with mobility as a key consideration, and therefore underperform under challenging mobile and wireless conditions. MobilityFirst, a clean slate architecture proposal, embraces several key concepts centered around secure identifiers that inherently support mobility and trustworthiness as key requirements of the network architecture. This includes a hop-by-hop segmented data transport that allows late and dynamic rebinding of endpoint addresses to support mobility.**

While this provides critical gains in wireless segments, some overheads are incurred even in stable segments such as in the core. Bypassing layer 3 decisions in these cases, with lower layer cut through forwarding, can improve said gains. In this work, we introduce a general bypass capability within the MobilityFirst architecture that could provide both better performance and enable both individual and aggregate flow-level traffic control. Furthermore, we present a detailed OpenFlow-based design to bypass layer 3 routing in MobilityFirst, using layer 2 VLAN tagging. Finally, we present a prototype that shows that it is possible to use OpenFlow to create the bypass.

I. Introduction

Mobile devices are becoming dominant in current networks and significant core architecture changes have been proposed to support them. Current protocols such as TCP/IP were not designed with mobility as a key design requirement. The inferior performance of these protocols in highly mobile networks and the increasing number of mobile devices has motivated the research community to design Future Internet architectures that consider mobility as a key design requirement [1], [2], [3].

MobilityFirst [1], [4] is a project funded by the NSF FIA program that designs a mobility-centric architecture for the future internet. MobilityFirst supports secure identifiers that inherently support mobility and trustworthiness. These mechanisms greatly enhance the support of mobile devices in the network. In the MobilityFirst architecture, data is transmitted between adjacent routers in a hop-by-hop manner. Entire chunks of data are received at the next hop before being forwarded again. Also, routing decisions are performed at each hop to ensure proper delivery if a node has disconnected and connected to another point of the network. However, this process also increases the delay needed to send data in a hop by hop manner [1].

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1040765.

Aravind Krishnamoorthy is currently working at Amazon. This work was done when he was a graduate student at Rutgers University.

Certain segments of the network are stable and allow exceptions to the storage and routing delays. If we know that a node will remain connected to the same access point for a period of time, we do not need to make routing decisions at every hop between the source and the destination. Also, segments within the core of the network are exempt of mobility requirements. In scenarios like this, it is possible to bypass the routing layer of MobilityFirst.

Bypassing MobilityFirst routers can improve the performance of the network, because the delay of forwarding data at a lower layer is smaller. Another advantage is that it enables flow aggregation. Multiple data transmissions can be encapsulated in the same flow. To illustrate the advantage of flow aggregation, imagine a football stadium with 80,000 users accessing resources on the Internet. Without any bypass, routing decisions will be made at each hop of the way between the source and the destination for each of the 80,000 users. If we assume that the users will remain in the stadium for a period of time, we can bypass the routing layer. It is very likely that the routes between the sources on the Internet and the destinations in the stadium share more than one MobilityFirst router. For those sections of the network that are shared, we can forward all the data using the same rule (informally, we can think of it as "Tag all traffic going to the stadium with VLAN 1". Once the data reaches the last hop of the bypass, each packet is routed to its specific destination accordingly. Therefore, using a small number of rules, we can forward the traffic intended for all the users in the stadium.

There are several ways of achieving this bypass. It could be done at layer 2 using VLAN tagging, or it could be done at layer 1 using either OTN tagging or WDM tagging. For any of the techniques used, there are challenges that have to be considered. Mobility, scalability, efficiency and reliability are four challenges that must be addressed by any bypassing technique.

In this paper, we propose using OpenFlow [5] to bypass the routing layer using VLAN tagging. OpenFlow is the most commonly deployed Software Defined Networking technology today. SDN decouples the data plane from the control plane in a network switch, by migrating the latter to a software based component. In an OpenFlow-based network, the controller can dynamically update the forwarding rules of a network device. The controller also has a centralized view of the network. Because of these capabilities, an OpenFlow-based network can be used to create, modify and delete layer 2 circuits to

bypass the routing layer. We have deployed MobilityFirst in an OpenFlow-based network using the ORBIT testbed [6] to experiment using OpenFlow in MobilityFirst.

In this manuscript, we begin by providing background information on MobilityFirst and an SDN-based implementation of MobilityFirst in section 2. In section 3, we describe how to bypass the L3 routing in MobilityFirst using L2 VLAN switching. In section 4 we explain how OpenFlow can be used to achieve the bypass using VLAN switching. In section 5, we show early results and in section 6 we discuss the conclusions and future work.

II. Background

A. Overview of MobilityFirst

The MobilityFirst project proposes a clean slate redesign of the Internet architecture [1]. This design aims at supporting mobile devices and applications as the main elements of the network. Cisco has predicted that by 2014, wireless devices will account for more than 60% of IP traffic. The current IP protocol was not designed for mobile applications and the emergence of such traffic offers an opportunity to evaluate what should be the purpose of functionality of the network [1].

Figure 1 shows the main building blocks of the MobilityFirst architecture. MobilityFirst provides three meta-level services: the global name resolution service (GNRS), the name-based services and the optional compute layer plug-ins. MobilityFirst also provides three core transport services: the hybrid GUID/NA global routing service, the storage aware routing (GSTAR) and the hop-by-hop transport [1]. In this background section we focus on explaining how routing is performed in MobilityFirst.

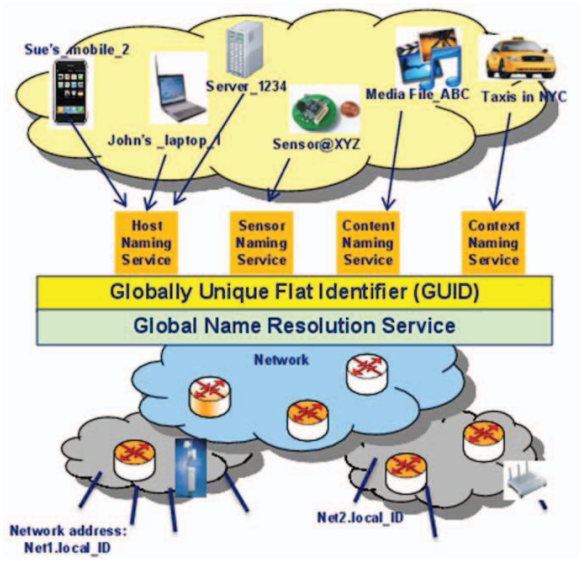

Fig. 1. Basic Protocol Building Blocks in MobilityFirst. Source: Raychaudhuri et al. [1]

1) Storage-assisted segmented data transport: MobilityFirst uses generalized storage-aware routing (GSTAR) [7]. In Fig. 1, suppose that a host wishes to send data to John's laptop. First the host should acquire Johns GUID. Then a packet is sent with the GUID as the destination. A MobilityFirst node resolves the GUID using the GNRS and obtains a list of NAs where the destination is connected to the network. The router sends a packet containing a destination GUID, a service identifier and a list of NAs. At each hop, a router will decide if the NA is within its reach or if the data must be forwarded to another router.

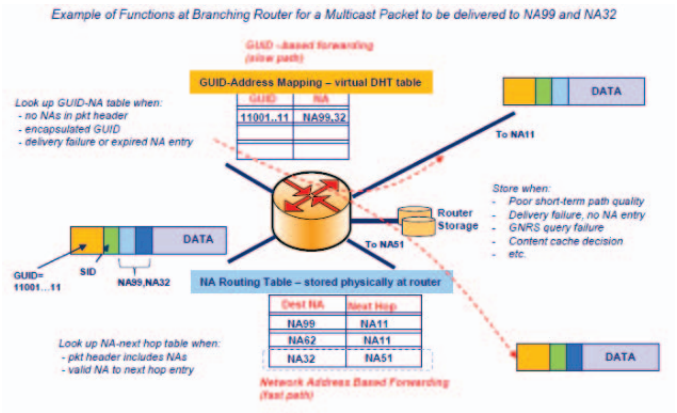

Fig. 2. Hybrid GUID/NA packet headers in MobilityFirst. Source: Raychaudhuri et al. [1]

MobilityFirst uses a hybrid name/address based routing to achieve scalability. The number of GUID objects is expected to be in the order of billions, but network addresses are expected in the order of millions. By mapping GUIDs to NAs, routing is greatly simplified [1]. Figure 2 shows how GUIDs and NAs are used during the routing process.

Another important feature of MobilityFirst is that the transmission of data in a hop-by-hop manner to support mobility. In this architecture, the entire file is received at each hop before transmitting it to the next one. Using this approach, it is possible to do storage-aware routing and late binding [1].

B. Software defined networking implementation of MobilityFirst

Software Defined Networking consists of decoupling the control and data plane of a network device. A software-based entity is responsible for the control plane. OpenFlow [8] is an SDN protocol that allows software applications to manipulate the flow table of a network switch. In this section, we briefly describe an SDN implementation of MobilityFirst using OpenFlow.

In a MobilityFirst network, data is split into entities called "chunks" before being transmitted. The size of a chunk can be anything ranging from MTU size of the link to large values like 64 MB or 128 MB. Each chunk is then made up of several packets (each packet being of the MTU size, 1500 bytes in case of Ethernet link). Suppose host1 wants to send a 5 MB

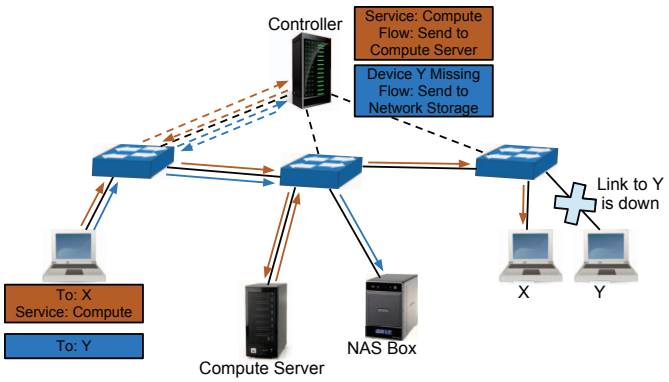

Fig. 3. MobilityFirst SDN architecture showing forwarding functions and in-network storage and compute service components.

file to host2. First, it splits the file into chunks (let's assume each chunk is 1 MB). So host1 now has 5 chunks, and each of those chunks has approximately 700 packets (of 1500 bytes each). When host1 transmits each chunk to MFRouter1, only the first packet of each chunk has the routing header (as in the destination GUID, service ID, etc.).

In our SDN implementation of MobilityFirst, the network controller is responsible for finding a path to transmit all the chunks from the source to the destination. When the first packet of each hop arrives to a switch, there is no forwarding rule for it. Therefore, the controller must perform several tasks. First, it must use the destination GUID of the packet to find the destination in the network. Second, it must compute which switch is the next hop of the path. Third, it must push a rule into the switch so that all the data of that chunk is forwarded to the next hop. This process is repeated for each chunk of data.

III. Bypassing L3 routing

In this section we discuss how to bypass layer 3 routing in MobilityFirst. First we describe the challenges of a bypassing technique. Next, we explain how to bypass the routing layer using layer 2 VLAN switching.

A. Challenges and design goals of a bypassing technique

Several challenges must be considered to bypass the routing layer in MobilityFirst: when to setup circuits and for how long; how many circuits are needed and their granularities and how to implement automated circuit creation in the MobilityFirst context.

- Mobility: It is important to keep in mind that nodes are assumed to be mobile. A circuit reservation solution cannot assume that a node will remain at the same location.
- Efficiency: The overhead of setting up circuits should be low and the circuits should significantly improve the performance of end-to-end deliveries.
- Scalability: The MobilityFirst architecture should be able to support a large number of users. The delay of setting up circuits must remain low for a large number of users and the number of circuits reserved should be able to scale as well.
- Reliability: A successful delivery must be ensured, even if a circuit exists and the node location changes or the link fails.

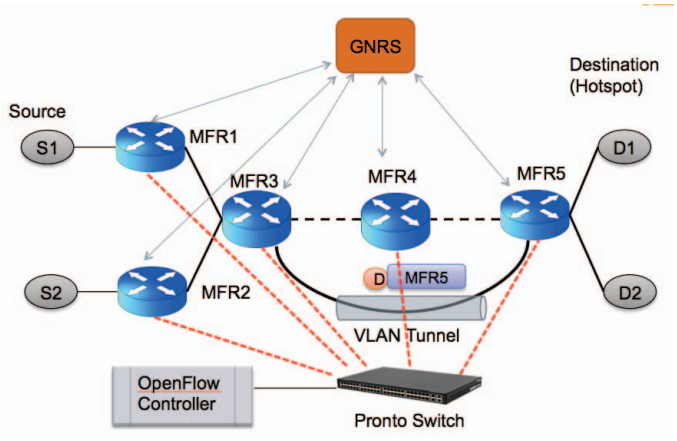

Fig. 4. Example of a bypass in MobilityFirst.

B. Bypassing L3 using Layer 2 VLAN switching

One way to bypass Layer 3 routing is to create Layer 2 circuits using VLAN tags. Recall that MobilityFirst works on a hop-by-hop basis. A MobilityFirst router sends the data to the next router and this is repeated until the destination is reached. Using this bypassing technique, a circuit can be created at L2 between the host and the destination. In order to do this, a path must first be found at the first hop to the destination. Next, a forwarding rule must be added in all forwarding elements so that the traffic is automatically forwarded to the next hop. To identify each flow, a VLAN tag can be used.

Figure 4 shows an example of a bypass. One source is attached to the MobilityFirst router 1 and another one is attached to the router 2. Since all destinations are attached to router 5, then a bypass between routers 3 and 5 can be created. Once the bypass is pushed, no routing operation is performed at router 4. The way to create this bypass is to add a forwarding rule to router 4 that forwards all traffic with a given VLAN from router 3 to router 5. In router 3, when we forward packets belonging to the bypassed flow (source is S1 or S2 and destination is D1 or D2), we tag them with the same VLAN number. When the data reaches router 5, routing decisions are taken based on the destination GUID of each packet. This ensures that different routes are chosen for destinations D1 and D2.

This design enables flow aggregation. In Fig. 4, a single rule in router 4 can be used to send data to multiple destinations. In any scenario where many destination nodes are connected to the same router, this feature is key to ensure the scalability of the system. In a more realistic topology, it is likely that end users are connected to edge routers and these devices are

interconnected through other devices across the network. Flow aggregation enables connecting multiple users connected to the same edge routers using a small number of rules. By reducing the number of rules needed at each hop, we significantly increase the scalability of the network.

As discussed earlier, this solution should also take mobility into consideration. If a circuit exists and a node changes the location, the delivery must still be guaranteed. If a bypass is in place and a node disconnects from the network, we must ensure that the current chunk of data is delivered to a MobilityFirst router that will find a new route. Also, subsequent chunks of data should not be sent through the bypass. In the example shown in Fig. 4, suppose the destination node D2 disconnects from router 5 and reconnects to router 4. When the data reaches router 5, it is still possible to locate node D2. By querying the GNRS about the location of the GUID of D2, we can learn that the location of the node has changed. Next, we can forward the packets to the next hop and we can also remove node D2 from the bypass.

This solution should be efficient as well. There is a trade-off between the time and resources that it takes to create a circuit to bypass L3 and the delay required at L3 routing. If a circuit is to be created, the time it takes to set it up should be significantly shorter than the time saved by bypassing L3. Also, the controller should require an acceptable amount of resources to detect when and how to create circuits. If the controller's performance is significantly decreased because of this, then the solution is not acceptable.

Finally, reliability must be taken into consideration. As we mentioned above, the delivery of the message must be guaranteed. If one of the links that are part of the bypass path fails, the data must be forwarded to a MobilityFirst router and the bypass must be deleted.

Another way to implement this traffic engineering technique would be to use multi protocol label switching (MPLS) [9]. Using MPLS, the ingress edge router computes the route from source to destination, communicates this route to all the routers involved and inserts a label into each packet. Successive hops can then forward packets based on the label. Note that this technique does not completely bypass the routing layer, as packets must still be processed by routers. In our approach, there is no need for the packets to be processed at the routing layer and all packets can be forwarded by simple L2 switches.

C. Deciding when to create a bypass

One of the major challenges of this implementation is deciding when to create a bypass. We envision two alternatives: proactive and reactive bypass creation. A proactive implementation is easier, but it requires that the nodes provide previous information. A MobilityFirst node could notify that a given number of bytes will be transferred to a destination. If this information is known, the controller can create a Layer 2 circuit between the sender and the receiver to ensure a faster communication. The advantage of a proactive approach is that the rules can be pushed in advance and the network controller does not need to make dynamic changes once the data starts flowing. However, a proactive solution only works when the information of the data transfer is known in advance, which is not always the case.

When no previous information is available, the bypass must be created in a reactive manner. In this case, the controller must dynamically identify for which flows to create a bypass. One possible approach is for the controller to store information about the location of devices. If multiple flows for a single destination are repeatedly forwarded to the same hop, the controller can assume that the node will not change the location for a period of time. Then, a bypass can be created for data sent to that device. The advantage of this approach is that it is completely dynamic and no previous information is required about the characteristics of the communication. On the other hand, the controller has to do more processing and this increases the delay. Also, the controller must store additional information and this can compromise the scalability of the solution.

D. Deciding when to remove a bypass

We also address how to remove a bypass. Once again, this can be done proactively or reactively. If a bypass was proactively created and we have information regarding when the data transfer will end, then the controller can automatically remove the bypass at a given time. However, a reactive solution must exist at any time, in case a disconnection happens. The controller can monitor which nodes get disconnected from the network. For each disconnected device, a clean way to remove the bypass is to maintain the flow rules for the current chunk, so that all the data of that chunk reaches the destination network device. However, for the next chunk, the standard data processing is applied and a hop-by-hop route is used.

IV. Implementation using OpenFlow

In this section we show how OpenFlow can be used to bypass L3 routing using L2 VLAN switching. We discuss how to push a circuit using OpenFlow and we discuss how we address the challenges mentioned in the previous section.

A. Mapping chunks to VLANs

We first describe some technical details of our OpenFlow-based implementation of MobilityFirst. In MobilityFirst, data is split in chunks and packets include information to know which chunk they belong to. For each chunk, the first packet is forwarded to the controller and a flow is pushed into the switch so that all the remaining packets of that chunk are forwarded to the next hop. To make this compatible with OpenFlow, the routing header is introduced in the L3 Source IP Address field. The controller can then parse the data of the first packet and use the routing information to compute where to forward all the packets of this chunk. When the next destination has been decided, a new flow rule is pushed to forward all the packets in this chunk to the next hop. To match all packets to the inserted rule, the hop ID is used as a VLAN tag. This hop ID identifies all packets belonging to one chunk across the link. Coming back to the example, for each of the five chunks, all

the 700 packets will have the same hop ID and this hop ID is also inserted as a VLAN tag in all the packets. If we use incremental hop IDs, then in the above scenario, all packets in chunk 1 will have hop ID 1, those in chunk 2 will have hop ID 2 and so on. This helps us identify which chunk a specific packet belongs to (since the packets themselves do not have any such information, except for the first packet of the chunk).

The key to achieve the bypass is to push a flow rule into all the switches between the source and the destination instead of only for one hop. In an OpenFlow-based network, the controller is aware of the topology. Thus, an end-to-end path can be found and all forwarding devices can be reached from the controller to push a new flow rule. To find a path between the host and the destination, we need to know the Layer 2 MAC address and the input and output ports at each hop. Next, specific flow entry rules can be pushed at each switch. The VLAN tag is the same for all the switches, but the source and destination MAC addresses and ports are different.

B. Floodlight controller

In the current OpenFlow-based deployment of MobilityFirst, the entire route between the host and the destination is computed using the service provided by Floodlight. We also implemented a mapper between GUID numbers and mac addresses. Given a GUID, the controller can find the MAC address associated to that node. Therefore, the information on the entire path is available. To achieve a bypass, we collect for each hop the following information: VLAN id, destination GUID, in-port and out-port. We push a flow rule into each switch using the proper port values and keeping the same VLAN id and destination GUID. As a result, all the packets of the current chunk are forwarded at layer 2 until they reach the final hop.

C. Discussion: Challenges addressed

We mentioned four key challenges for the bypassing technique: mobility, efficiency, scalability, reliability. Next we discuss how our solution addresses those points and what are the challenges that must still be overcome.

Our solution addresses mobility by routing packets at the end of the bypass. If a bypass goes from router 3 to router 5, then the data will be received at router 5 and a route will be computed for the GUID or NA. If a node has connected to a different location of the network, the controller can query the GNRS for the new NA and find a new route. We do not expect this scenario to occur on a regular basis, because a bypass should be pushed only when a node is not expected to move. However, if the device does move, a new route can always be found. One challenge that remains is to actually be able to push bypasses only when the nodes will remain in the same location. Otherwise, the delay introduced can become significant.

In terms of efficiency, OpenFlow is a convenient approach to dynamically manipulate forwarding rules. The application running on the controller can proactively or reactively modify the flow table of one or more switches. Therefore, creating or deleting a bypass can be done efficiently. If a bypass is created proactively, the controller only needs to act at a specific time. If the bypass is to be created reactively, the controller must incur a delay to process the first packet of each chunk to decide if a bypass is needed or not. We expect this delay to be acceptable, as only the first packet of a chunk must be processed. However, an interesting scenario occurs when there is a failure during the transmission and the distance between the start and end of the bypass is far. In this case, the time to send the contents again can introduce an important delay. This raises the question of whether to bypass a large number of hops or if it is more convenient to keep the number of hops small.

Regarding scalability, we discussed earlier how flow aggregation can help the network scale. Using OpenFlow, we can easily update any flow entry of a device. If a bypass already exists, the controller can easily modify the rule so that the bypass includes a new source or a new destination. If a bypass must be created, it can be done efficiently too. Finally, the fact that the controller has a centralized view of the network allows the application to be aware of changes in the topology fairly quickly. This simplifies updating a bypass when necessary. On the other hand, the limited number of VLANs and the size of the flow table are known limitations in an OpenFlow-based network. It is important to evaluate if these limitations significantly impact the scalability of this deployment.

Finally, our solution also addresses reliability because the architecture is still storage-aware. In MobilityFirst, the data is stored at each hop before being transmitted. If the controller detects that the data is not properly delivered to the destination router at the end of the bypass, it can use a hop-by-hop delivery. It is important to evaluate how often does this occur in real-life scenarios, in order to measure the impact on the performance of the network.

D. Discussion: Centralized control plane

One key feature of OpenFlow-based networks is that the control plane is centralized. The advantage of a centralized control plane is that the controller has a network-wide knowledge of the network. This simplifies reacting to failures and creating new paths when necessary. The OpenFlow protocol includes features that allow a controller to listen to switch events and thus learning about broken links and connected devices. The main drawback of a centralized control plane is the scalability challenge, as well as becoming a single point of failure. To overcome this, distributed control plane architectures such as HyperFlow [10] and ONOS (Open Network Operating System) [11] have been proposed.

V. Experimental results

In this section, we experiment with a proactive bypass creation and removal, and we present our results.

A. Experimental setup

In order to demonstrate the feasibility of the bypass, we simulate a MobilityFirst network using Mininet [12]. We

TABLE I
ROUND TRIP TIME IN MILLISECONDS WITH AND WITHOUT A BYPASS.

	First packet	Rest of the packets (average)
With bypass	0.091	0.09
Without bypass	58.4	0.15

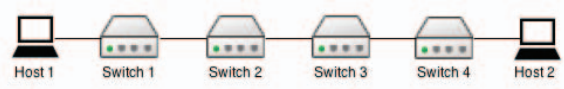

Fig. 5. Topology of the experiment.

deploy a linear topology with four switches. We measure the rountrip time between hosts 1 and 2 when a bypass exists and when it does not. To do this, we use the ping tool provided by Mininet. For all our experiments, we use one single chunk of data (to avoid sending VLAN tagged packets from Mininet). We send 30 ICMP requests from host 1 to host 2.

In the first experiment, we measure the round trip time of a ping request between host 1 and host 2 when no bypass is pushed. In this scenario, all the data is sent to each hop. At each hop, no rule exists in the switch and the first packet is sent to the controller. The controller performs layer 3 routing and pushes a rule into the flow table. The rest of the packets are matched to this rule and are forwarded at layer 2.

In the second experiment, we measure the round trip time of a ping request between host 1 and host 2 when two bypasses are proactively pushed. One bypass is from host 1 to host 2 and the second one is from host 2 to host 1. In this scenario, the controller does not receive any packet in, because all packets are matched to rules that were proactively pushed.

B. Results

Table I shows the results with and without a bypass. As expected, when a bypass does not exist, the first ping request encounters a significant delay (58 ms) during the round trip time from host 1 to host 2. In average, the rest of packets are forwarded in under 1ms. The goal of these experiments is to demonstrate how the bypass if feasible and how it significantly reduces the forwarding time of the first packet. More complete experiments in the ORBIT testbed to evaluate the performance and scalability of the solution are left as future work.

Notice that this experiment is only for one chunk of data. If multiple chunks were sent, the first packet of each chunk would face a similar delay. This shows that creating a bypass significantly reduces the transmission time of the first packet of a chunk. The results also show that, for the remaining packets of a chunk, the round trip times are similar with or without a bypass.

VI. CONCLUSION AND FUTURE WORK

This paper discusses how to bypass layer 3 routing in MobilityFirst. The advantage of such a bypass is to eliminate the delay introduced at that layer. We discussed how to use OpenFlow to bypass layer 3 routing in MobilityFirst using layer 2 VLAN tagging. Instead of pushing a flow rule to one switch only (as it would be done to ensure a hop-by-hop communication), we push rules into all the switches of the path between the source and the destination. By doing this, we ensure that all data is forwarded at layer 2. We also discussed how this technique enables flow aggregation. By managing several data transfers using a small number of forwarding rules, we increase the scalability of the network.

Our experimental results, obtained using Mininet, show that the first packet of a chunk encounters significant delays when there is no bypass. When a bypass exists, all packets are transmitted in a much smaller time. These results are a proof-of-concept that show that OpenFlow can be used to bypass the routing layer using VLAN tagging. Future work includes experimenting on the OpenFlow-based version of MobilityFirst available on ORBIT. These experiments will allow us to test the performance issues and to evaluate the scalability of the network. We identified a series of challenges that must still be considered, such as the impact of mobility or the devices and failures during the transmission when a bypass exists. We intend to work on these evaluations for more realistic scenarios.

REFERENCES

[1] D. Raychaudhuri, K. Nagaraja, A. Venkataramani, "MobilityFirst: A Robust and Trustworthy Mobility-Centric Architecture for the Future Internet," in *IEEE Computer Communications Workshop*, October 2010.
[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 2009, pp. 1–12.
[3] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proc. ACM SIGCOMM*, August 2008.
[4] "MobilityFirst," http://mobilityfirst.winlab.rutgers.edu/.
[5] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–20, 2013.
[6] OpenFlow Experimentation in ORBIT. [Online]. Available: http://www.orbit-lab.org/wiki/Documentation/OpenFlow
[7] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "GSTAR: Generalized storage-aware routing for mobilityfirst in the future mobile internet," in *Proc. of MobiArch*. ACM, 2011, pp. 19–24.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[9] Umesh Lakshman and Lancy Lobo, "MPLS Traffic Engineering," http://www.ciscopress.com/articles/article.asp?p=426640.
[10] A. Tootoonchian and Y. Ganjali, "HyperFlow: a distributed control plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, August 2010.
[11] On.Lab, "OpenSource SDN Stack," http://onlab.us/tools.html.
[12] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.