Mobility Management Framework in Software Defined Networks

Kyoung-Hee Lee
Department of Computer Engineering, Pai Chai University, Korea
leekhe@pcu.ac.kr

Abstract
There have been quite a number of approaches for mobility management in IP networks to provide service continuity to mobile nodes traveling across heterogeneous wireless networks. One of the reasons making those existing approaches difficult to be applied into real network environment is that they commonly require significant functional updates on network devices including routers, control servers and mobile terminals. The complexity to manage huge mobility information and much overhead to exchange mobility control messages also became obstacles to deploy those approaches. Recently, software defined networking is being considered as a dynamic and scalable architecture which can resolve those limitations by decoupling network control and data forwarding functions, and enabling the network control to become directly programmable in a centralized manner. In this paper, we address some issues and possible solutions to introduce a well-known IP mobility management approach, PMIPv6, into software defined networks. A feasible MM framework is also proposed with consideration of efficiently utilizing the features and advantages of the software defined networking architecture.

Keywords: Mobility Management, PMIPv6, Software Defined Networking

1. Introduction
For a couple of decades, mobility management (MM) techniques in IP networks have been got a lot of attention as fundamental solutions to provide service continuity to mobile users moving around various wireless networks. There have been lots of MM approaches, including Mobile IP (MIP) [1-2] and Proxy MIP (PMIP) [3], which were carefully designed and proposed to give an efficient framework and practical protocols to address this issue. Those approaches commonly require some special network entities to manage location information and to control handover situation of mobile nodes (MNs), for example, called Localized Mobility Agent (LMA) and Mobility Access Gateway (MAG) in PMIPv6. Those new functions should be implemented either on additional control servers or numerous access/gateway routers in all over the existing networks. It requires replacing a huge number of network devices or sweeping functional updates on the current network infrastructure. This would be one of the major reasons making those MM approaches difficult to be deployed into the real networks.

Another critical problem could be the complexity to manage all mobility information of a tremendous number of MNs and the message overhead to control frequent handovers of them. A location of an MN changes often as it moves continuously, and routing paths for the MN should be updated in time from/to all corresponding nodes communicating with the MN. Those updates should be done at all relevant routers and switches scattered in the networks as well as mobility control servers. These heavy MM procedures may degrade the speed and utilization of networks but the network service providers have not found proper commercial
merits to endure the degradation even though it is evident that the MM service could provide their customers with an innovative networking environment.

Recently, software defined networking (SDN) architecture emerges as a key solution to overcome complexity caused by thousands of protocols and control functions implemented in the current network infrastructure and to provide network operators with a managed and programmable control plane separated from the underlying data-forwarding plane [4]. In the SDN architecture, most of networking functions, including routing, management, mobility, quality of service (QoS) and access control, can be implemented as software in the control plane which does not require computational overhead and resources in switches or routers residing in the data-forwarding plane. This feature may enable some enhanced networking services such as mobility support among various wireless networks to be introduced into the real networks covering wide areas.

In this paper, we consider how to deploy a well-known MM approach, PMIPv6 [3], into the SDN architecture. A feasible framework to efficiently implement PMIPv6 over SDN is proposed accordingly. Some relevant issues or problems are introduced, and the possible solutions are also proposed.

2. Related work

2.1. SDN Architecture

According to Open Networking Foundation (ONF), SDN is defined as an architecture decoupling the network control and forwarding functions to enable the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [4-5]. Figure 1 shows a logical view of SDN architecture.

![Figure 1. Logical View of SDN Architecture](image)

As shown in the figure, most network services (i.e., protocols and controls) are centralized at SDN control software. The SDN controller manages all network devices in the underlying infrastructure layer, which are virtualized a single logical switch. Network operators can control the networks through standard interfaces (e.g., OpenFlow) independently of network device vendors. The network devices are only equipped with data-forwarding functions controlled by the SDN controller. This makes design and operation of a network simpler and more efficient. APIs between application layer and control layer can provide a virtualized
network environment and means to apply various polices regarding routing, access control, traffic engineering, power management, etc.

2.2. OpenFlow

OpenFlow is a key element of the SDN architecture, which is an ONF standard protocol to remotely control the forwarding tables of switches or routers [9]. OpenFlow controller, a part of SDN controller, gives instructions through secure channels to OpenFlow switches which are generally layer 2 (L2) switches with OpenFlow client software. An OpenFlow switch manages multiple flow tables to handle and forward packets to their destinations. Figure 2 shows the OpenFlow protocol architecture introduced in a survey paper on SDN [10].

![Figure 2. OpenFlow Protocol Architecture](image)

An entry in flow table consists of a rule (or match fields) used to match incoming packets, consequential packet handling actions and statistics (i.e., a counter of packets matched to the rule). Match files may contain information found in the packet header, ingress port and metadata. Actions, a set of instructions, are to be applied upon a match and dictating how to handle those matching packets. A counter is used to collect statistics for the particular flow such as the number of received packets. An OpenFlow switch make, update and delete the flow entries in its flow tables according to instructions delivered through OpenFlow protocol from the OpenFlow controller.

2.3. RouteFlow

RouteFlow is one of deployment frameworks for SDN, which is being developed as an open source project [11]. This approach runs IP routing protocols (e.g., BGP and OSPF) at a centralized RouteFlow server and generates the forwarding information base (FIB) according to the configured routing protocols. The server collects IP and ARP tables to be translated into OpenFlow rules which are finally installed in the associated programmable switches in the networks [12].

Figure 3 shows the architecture of RouteFlow approach described in [12]. A RouteFlow server generates and operates virtual machines (VMs) in the SDN control plane. A VM in the RouteFlow server is mapped to a particular OpenFlow switch in the data-forwarding plane. VMs build a virtual topology and run an open source routing protocol, Quagga [13-14]. The RouteFlow server continuously monitors the status of routing tables in VMs. If a change is
found from the routing tables, the RouteFlow server makes a corresponding flow entry and delivers it to a relevant OpenFlow switch immediately.

![Figure 3. RouteFlow architecture](image)

The RouteFlow architecture illustrates a specific and efficient direction to accommodate IP routing protocols in the SDN control plane. The following chapters of this paper address frameworks to apply two well-known IP-based MM approaches into the SDN architecture based on the RouteFlow approach.

3. PMIPv6 over SDN

A demand of network-based control to mobility support has been continuously increased to minimize the functional changes on a user terminal device. PMIPv6, which is a standard MM framework developed by IETF, defines a Localized MM (LMM) domain and specifies the architecture and protocol to handle mobility within an LMM domain with a network-based manner [3, 15].

In the PMIPv6 architecture, an LMA and several MAGs constitute an LMM domain. An MAG initiates and executes a layer 3 (L3) handover procedure for an MN. The LMA manages all MN’s location information while they are moving among MAGs in a single LMM domain. A data tunnel is established between the LMA and each MAG to forward packets for MNs. When an MN enters a particular MAG’s area, the MAG registers its own IP address with the LMA as the MN’s care-of-address (CoA). The home address (HoA) of the MN does not change in a single LMM domain. By managing the binding information of the MN’s HoA and CoA, the LMA can properly forward packets from the outside of the LMM domain to MNs.

Figure 4 illustrates an MM framework deploying PMIPv6 into the SDN architecture based on RouteFlow. The network topology in the figure consists of a single LMM domain. Three VMs in the control plane are running the PMIPv6 protocol to act as an LMA and two MAGs, respectively. VM #1 plays a role of the LMA located at the gateway router and it is logically mapped to Switch #1 (SW #1). VM #2 and VM #3 are two MAGs running at access routers (ARs), which are mapped to SW #2 and SW #3, respectively. All switches are assumed to be OpenFlow switches. Flow tables on those switches are managed by the SDN controller (i.e., RouteFlow server) according to the status of routing tables in VMs. Bidirectional IP-in-IP tunnels are established among those switches to deliver packets from/to MNs.
The operations of PMIPv6 in the control plane are not different from those of the existing PMIPv6 standard. If an MN moves from the SW #1’s area to the SW #2’s area, MAG2 in VM #3 initiates an L3 handover procedure for the MN. MAG2 sends a Proxy Binding Update (PBU) message to the LMA in VM #1 to notify a change of the MN’s location. Then the LMA updates its binding cache so that MAG2 represents the MN’s CoA. After updating the binding cache, the LMA replies to MAG2 with a Proxy Binding Acknowledgement (PBA) message. The RouteFlow server translates the changes in the LMA’s binding cache into OpenFlow rules. These rules are immediately installed to SW #1 through OpenFlow protocol. Thereafter, SW #1 forwards incoming packets destined to the MN toward a tunnel established to SW #3.

<table>
<thead>
<tr>
<th>Rules</th>
<th>Actions</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  IP dst = MN, incoming port = P1</td>
<td>Encapsulate packet (from LMA to MAG2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forward packet to port P3</td>
<td></td>
</tr>
<tr>
<td>2  Packet encapsulated (from MAG1 to LMA)</td>
<td>Decapsulate packet (from LMA to MAG2)</td>
<td></td>
</tr>
<tr>
<td>IP dst = MN</td>
<td>Encapsulate packet (from LMA to MAG2)</td>
<td></td>
</tr>
<tr>
<td>IP src = MN</td>
<td>Forward packet to port P3</td>
<td></td>
</tr>
<tr>
<td>3  Packet encapsulated (from MAG2 to LMA)</td>
<td>Decapsulate packet</td>
<td></td>
</tr>
<tr>
<td>IP src = MN</td>
<td>Forward packet to port P1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rules</th>
<th>Actions</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  IP src = MN, incoming port = P2</td>
<td>Encapsulate packet (from MAG2 to LMA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forward packet to port P1</td>
<td></td>
</tr>
<tr>
<td>2  Packet encapsulated (from LMA to MAG2)</td>
<td>Decapsulate packet</td>
<td></td>
</tr>
<tr>
<td>IP dst = MN</td>
<td>Forward packet to port P2</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4. PMIPv6 Architecture over SDN**

**Figure 5. Flow Tables in OpenFlow Switches**

Figure 5 depicts a simple example of flow tables in the OpenFlow switches to support mobility of the MN visiting the SW #3’s area as shown in Figure 4. If a packet arrives from outside of the LMM domain and its destination address points to the MN, SW #1 encapsulates
and forwards the packet to MAG2 through port 3 (Rule 1 of SW #1). When receiving the packet, SW #3 decapsulates it and forwards the inner packet to the MN through port 2 (Rule 2 of SW #3). On the other hands, a packet sent from the MN arrives at port 2 of SW #3. Then SW #3 encapsulates and forwards the packet to SW #1 through port 1 (Rule 1 of SW #3).

4. Considerations on the Proposed Framework

One of open issues on deploying PMIPv6 into the real networks is how an MAG quickly knows an entry of an MN which has just moved from another MAG’s area. Because an MAG initiates an L3 handover procedure of PMIPv6 for an MN, how rapidly it knows the MN’s visit is quite important in the aspect of performance on handling handover situations. The PMIPv6 standard does not restrict what could be a trigger to instruct an MAG to send a PBU message to an LMA [3]. This issue can be more serious problem when PMIPv6 is deployed over the SDN architecture since an MAG at the control plane is separated from a switch in the data-forwarding plane, which is physically connected to MNs.

A reasonable solution is to introduce cross-layer functionality between L2 and L3 layers. An L2 event that a MN changes its point of attachment (PoA) can be useful information for upper layer protocols to efficiently control L3 or higher layer handovers. IEEE 802.21 standard, Media Independent Handover (MIH) service, provides such functions enabling upper layers to precisely know the underlying link status and behaviors [16]. Those functions do not support only local cross-layer interactions at a single node but also serve the remote interactions among different nodes. There have been some network-based MM approaches accelerating handover control with those L2/L3 interaction functions [6-8]. This idea could be employed by the proposed framework to reduce handover latency affected by the L2 handover notification time. A weakness of this solution is all PoAs and MAGs should be equipped with additional functions for L2/L3 interaction. An MAG’s functions could be easily updated because it is implemented at a VM. But it is not a simple requirement to add new functions to all type of PoAs used for various wireless access networks.

Another possible solution is to use Internet Control Message Protocol (ICMP), which has been proposed by the PMIPv6 standard [3]. According to the scheme, an MN broadcasts an ICMP Router Solicitation (RS) message after attaching a PoA located in a new MAG’s area. This RS message is delivered to the MAG and could be a trigger to start an L3 handover procedure. This process is basically performed by an MN to solicit an IPv6 prefix to be used in a new network. Thus the advantage of network-based control is kept while requiring no additional functions on MNs. A drawback of this approach could be a handover latency may increase according to the time that an MN’s current IPv6 prefix is expired and it sends an RS message to get a new prefix.

The PMIPv6 control message flows in [3] need be slightly modified when assuming the underlying network is based on the SDN architecture. Because an MAG is located inside of a VM in the control plane, an MN’s RS message cannot be directly delivered to the MAG. An OpenFlow switch mapped to the MAG should encapsulate the RS message and forward it to the MAG. A reply to the MN, a Router Advertisement (RA) message from the MAG, should be also encapsulated and delivered through the OpenFlow switch. Figure 6 shows a modified message flow according to this consideration.
Another issue on the proposed PMIPv6 architecture over SDN is concerned in its tunneling scheme. PMIPv6 establishes a data tunnel between an LMA and each MAG to forward packets for MNs residing in the LMM domain. This concentrates overhead to manage all MN’s location information on the LMA and enables MAGs to handle packets for MNs more simply. On the other hand, the LMA may suffer from excessive data traffic since all packets from/to MNs definitely have to path through the LMA. Anyway, this tunneling scheme looks suitable for PMIPv6 deployment in legacy networks. A question could be the tunneling scheme is still efficient when we assume the underlying networks are SDNs. An OpenFlow switch may be equipped with tunneling functions. If packets from/to lots of MNs share a single data tunnel, an MAG’s burden would decrease due to escape of individual routing for the packets. However, to maximize the benefits of SDN architecture, a switch needs to mainly perform data-forwarding functions such as flow switching, VLAN switching, etc. An OpenFlow switch may have a group table which a group entry consists of a Group ID of multiple flows and a set of corresponding actions (i.e., action bucket) [9]. The advantage from sharing a single data tunnel can be replaced with the effect of group table by allocating the same Group ID to multiple MNs and putting their common actions into the action bucket. Thus the overhead from packet encapsulation/decapsulation of tunneling scheme may not need to be endured any more in the SDN architecture.

In the PMIPv6 network, to exchange packets between two MNs which are located in a single LMM domain but in different MAG’s areas, those packets should pass through an LMA and two data tunnels. It may cause a bottleneck problem at the LMA while a lot of MNs are visiting the LMM domain and they exchange much traffic with each other. This problem can be mitigated if we assume that the tunnels are not mandatory. Because an LMA knows a list of all MNs staying in its domain, it can easily install rules to the underlying data plane so as to handle those internal packets to be delivered along the shortest path among MAGs. Thus a switch mapped to the LMA can escape from burden to handle those internal packets.

Above aforementioned issues, there remain several considerations to make the proposed PMIPv6 architecture over SDN more feasible to be deployed into real networks. Massive mobility information will be concentrated at a single node, the RouteFlow server, since all LMA and MAGs are running in VMs in the RouteFlow server. How to efficiently distribute MM functions of a single server into the data center needs to be carefully studied to address this issue. It is also to be determined whether the growth of flow tables in OpenFlow switches.

![Modified PMIPv6 Message Flow over SDN](image-url)
is affordable when the number of MNs considerably increases and they frequently moves across the networks. How the distance between the RouteFlow server and OpenFlow switches affects the handover control latency would be another important research issue in the future.

5. Conclusions

In this paper, we introduced an idea that SDN technology can be used to build a proper environment to accommodate the existing MM approaches. The SDN architecture decouples network control and data forwarding functions and enables the network control to become directly programmable in a centralized manner. These features may solve the limitations of the existing MM approaches, which they commonly require significant functional updates on network devices including routers and switches.

We proposed a framework to deploy a well-known MM protocol, PMIPv6, into the SDN architecture. The proposed MM framework is based on the RouteFlow approach so that all functions of LMA and MAGs are implemented in VMs of the RouteFlow server in the control domain. This enables an OpenFlow switch in the network to properly handle packets for MNs without any functional changes for mobility support. It was presented how flow tables in OpenFlow switches can be managed according to PMIPv6 operations. We addressed some considerations on the proposed PMIPv6 architecture over SDN, including L3 handover triggering, data tunneling and intra-domain packet delivery, with possible solutions on them.

For future work, how to enhance the proposed framework by distributing burden to handle massive mobility information into a cloud data center will be carefully studied.

Acknowledgements

This work was supported by the research grant of Pai Chai University in 2014.

References


Author

Kyounghee Lee, he received his M.S. and Ph.D. degrees in Korea Advanced Institute of Science and Technology (KAIST) in 2000 and 2006, respectively. From 2006 to 2013, he worked for Electronics and Telecommunications Research Institute (ETRI). Since 2013, he has been a faculty member of Department of Computer Engineering in Pai Chai University. He has expertise in international standardization on mobility management technology area as a Rapporteur of Q.22/SG13 in ITU-T. His research interests include mobile communications, Internet QoS, real-time multimedia service and cloud computing.