15

Software Defined Mobility Management for Mobile Internet

Jun Bi and You Wang
Tsinghua University, Beijing, China

15.1 Overview

This chapter proposes to use software defined networking (SDN) to address mobility management in the Internet. This chapter first reviews existing mobility protocols in the Internet and points out their drawbacks. Then this chapter explains why SDN is a promising way to solve these problems, followed by a description of SDN-based mobility management architecture for mobile Internet. This chapter also presents an instantiation of this architecture that is designed and implemented using OpenFlow, as well as related evaluation and comparison with existing Internet mobility solutions to illustrate the advantages of the proposal.

15.1.1 Mobility Management in the Internet

Internet mobility has been an active research topic for over two decades. Along with the evolution of the Internet, especially the growing of mobile data due to more and more mobile devices and applications, many research efforts have been paid to address Internet mobility. However, so far, there is no consensus on how to provide mobility support in the Internet, making it remain an open issue.

15.1.1.1 Mobility Management in the Internet and Cellular Networks

Internet mobility research is different from that in cellular networks. Although cellular networking has been providing mobility support to global users, it may not replace the role of Internet mobility support because of their disparate bandwidths, costs, service models, etc. [1, 2]. Moreover,

Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture, First Edition.
Edited by Madhusanka Liyanage, Andrei Gurtov, and Mika Ylianttila.

mobility in the Internet is showing new features comparing with that in cellular networks, that is, Internet mobility refers to not only movement from one point of attachment to another but also inter-ISP handover and even interdevice switching.

On the other hand, mobility management research in the Internet and cellular network is closely related, especially after IP is considered as the core part of future cellular networks [3]. Since the evolution trend of cellular networking is moving toward an all-IP-based infrastructure, which means all traffic leaving base stations becomes IP based and is delivered over packet-switched networks, IP mobility management becomes a key role to support future wireless systems.

Many IP mobility solutions serve as candidates to provide mobility management in cellular networks [4, 5], and some proposals have already been integrated into cellular networking. For example, Proxy Mobile IPv6 (PMIPv6), which is a typical IP mobility solution, has been adopted by the cellular core network Evolved Packet Core (EPC) in the 3rd Generation Partnership Project (3GPP) [6]. Besides, cellular backhaul technologies are also evolving toward IP-based designs, such as femtocells, which deploys home-located cellular base stations to provide connectivity to cellular users over their IP networks [7].

Current and future researches on Internet mobility management will continue to contribute to cellular networking. Recently, along with the development of 3GPP Long-Term Evolution (LTE), there is a growing trend to provide a more flexible and dynamic mobility management, which is also a concern in the Internet research area. The Internet Engineering Task Force (IETF) has already been standardizing related protocols, which may be introduced into 3GPP to replace its existing mobility management functions [8]. Due to the same reasons, we believe that this chapter also offers beneficial references for developing mobility management systems in current and future cellular networks.

15.1.1.2 Existing Internet Mobility Solutions

Generally, supporting Internet mobility means to offer uninterrupted Internet connectivity to mobile nodes (MNs) (mobile devices, users, or other entities), which change their attachment points while roaming in the network. Internet mobility is difficult to realize because it was an unforeseen feature when the Internet was built, and unfortunately, this feature contradicts with the current Internet architecture: due to the tight coupling of TCP and IP [9], changing IP addresses will cause interruption of TCP sessions on MNs, which may seriously impact experiences of mobile users.

Basically, Internet mobility solutions can be divided into two categories, that is, routing-based approach and mapping-based approach [10]. Routing-based approach makes an MN use the same IP address while roaming and thus requires dynamic routing to keep reachability of the MN. On the contrary, mapping-based approach allows an MN to change IP addresses but keep a piece of stable information, known as *identifier*, which does not change during movement. To reach the MN using its identifier, a mapping mechanism is introduced to resolve identifier to the MN's current *locator* (normally represented by its IP address). In these solutions, TCP sessions are always bound to identifiers instead of IP addresses; thus, they can keep survivability facing changing IP addresses. As discussed by Zhang et al. [1], routing-based approach is not suitable to provide mobility support in the global Internet, because the whole network requires to be informed of each MN's movement that may not scale well in large networks. Therefore, this chapter focuses on mapping-based approach.

Among all related proposals, Mobile IP (MIP) [11, 12] and its extensions [13, 14] are the earliest and most well-known protocols. MIP is an IETF-standardized protocol that allows MNs to keep session survivability while roaming around and changing IP addresses. Later, a large number of MIP derivatives have been proposed to improve its basic functionality [15–18]. Recently, there are also many individual IP mobility protocols [19–22] as well as future Internet architectures [23–28] that arise to address mobility problems in the Internet.

15.1.2 Integrating Internet Mobility Management and SDN

From one point of view, the key of providing mobility support in the Internet is to properly distribute the MN's identifier-to-locator mapping within the network so that its correspondents can reach the MN directly or indirectly. Although there exist various ways to realize such a function, they have drawbacks in different aspects, making it remain an unsolved issue. This chapter tries to address the problem using SDN and OpenFlow. SDN is an emerging network architectural approach, while OpenFlow is one of the most well-known instantiations of SDN [29]. In SDN, network structures, functions, and performance can be defined in a simpler way, which is usually achieved by providing programmable devices and a centralized control logic. As will be shown in this chapter, network functions or services required to support IP mobility can also be realized in a software defined way.

SDN helps to solve problems in IP mobility protocols for the following reasons: firstly, programmable SDN devices enable the *flexibility* of SDN-based mobility solution, which is a lack of existing IP mobility solutions. Specifically, the mapping of each MN can be flexibly placed on any SDN device instead of fixed Home Agents (HAs) or CNs, and this feature provides the basis to make SDN-based mobility solution become *adaptive* to diverse mobility scenarios. Secondly, centralized control let SDN-based mobility solution be aware of all kinds of mobility details, for example, how the MN moves, how the CN-to-MN traffic flows, etc. These details help to generate *optimal* strategies to handle different mobility scenarios via a lightweight algorithm without introducing complex distributed protocols. Thirdly, IP mobility in SDN architecture requires less host involvement. Most mobility functions can be realized on the network side as will be shown in Section 15.3, and this implies faster handoff without IP reconfiguration, less signaling overhead especially on wireless links, as well as higher security and privacy assurance.

15.1.3 Chapter Organization

The remainder of this chapter is organized as follows: Section 15.2 gives a classification and overview of related Internet mobility solutions. To further make clear of the problem this chapter addresses, Section 15.2 also presents a study on the mobility management functions of related solutions and discusses on their pros and cons. Section 15.3 proposes an SDN-based mobility management architecture for mobile Internet, together with an instantiation of this architecture that is designed and implemented using OpenFlow as well as performance evaluations and experiments.

15.2 Internet Mobility and Problem Statement

This section gives an overview of Internet mobility solutions and then discusses on the drawbacks of these solutions to further make clear of the problem this chapter focuses. Finally, a brief discussion on addressing Internet mobility in an SDN way is presented. Detailed protocol design, implementation, and evaluation of the solution this chapter proposes are given in the next section.

15.2.1 Internet Mobility Overview

One of the earliest Internet mobility solutions is MIP, which began its standardization in the IETF about two decades ago. Since then, various MIP derivatives have been proposed to improve the original protocol in order to adapt to the evolving Internet. Another category of solutions mainly relies on end hosts to realize mobility management. These protocols belong to "Identifier/Locator Split" (ILS) designs. ILS is an architectural model that points out that IP address has embedded both identifier and locator semantics and a split of the two is necessary. The concept of ILS had also been discussed many times during the past two decades and currently has got wide acceptance [30–32].

Besides MIP and ILS solutions, many future Internet architecture proposals also try to provide Internet mobility support. However, future Internet architecture proposals are usually clean slate and require substantial changes to the current Internet. They even do not rely on IP to work, which makes it difficult and inappropriate to compare them with current IP-based mobility solutions. Therefore, this chapter does not go further into mobility management proposals in future Internet architectures.

15.2.1.1 MIP and Its Derivatives

MIP derivatives are based on two origin protocols: MIP [11] and Mobile IPv6 (MIPv6) [12]. The core idea of MIP is illustrated in Figure 15.1a by taking MIPv6 as an example: the protocol uses a special type of IP address called Home Address (HoA) to identify an MN. When an MN moves to a new network, it obtains a Care-of Address (CoA), which can be used to reach the MN. Then it communicates with the HA located in its home network to update the binding cache that maps the MN's HoA to its current CoA. Since a CN does not know the MN's CoA, it sends packets to the MN using its HoA; thus, the packets are forwarded to the HA. With up-to-date binding cache, the HA can then encapsulate and redirect packets toward the MN's current CoA.

MIP centralizes both mobility signaling and data forwarding functions into a single HA, which increases signaling cost when MN is not within the home network. To address the problem, some extensions to MIP are proposed. Hierarchical Mobile IPv6 (HMIPv6) [13] deploys Mobility Anchor Points (MAP) in the network and uses them to localize mobility signaling when the MN is away from HA. Specifically, MN attaches to a nearby MAP, which is located using a Regional CoA (RCoA), and then the MAP is responsible for keeping the bindings between the MN's HoA and a Local CoA (LCoA), which is exactly the MN's current location, and tunneling packets to the MN. When attaching to a new MAP, MN informs HA of the new MAP's RCoA to keep reachability of the MN. PMIPv6 [14] is a similar solution, and

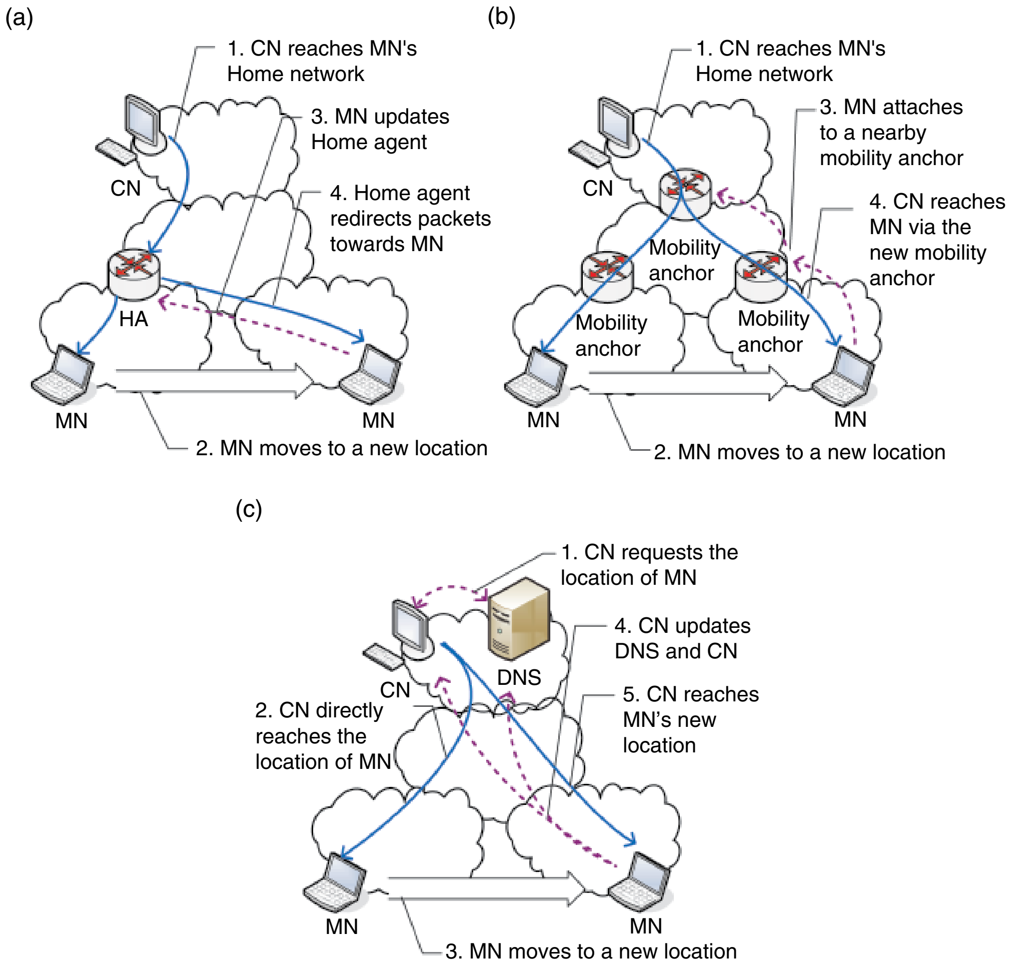
(a)
1. CN reaches MN's Home network
3. MN updates Home agent
4. Home agent redirects packets towards MN
CN
HA
MN
MN
2. MN moves to a new location
(b)
1. CN reaches MN's Home network
3. MN attaches to a nearby mobility anchor
4. CN reaches MN via the new mobility anchor
CN
Mobility anchor
Mobility anchor
Mobility anchor
MN
MN
2. MN moves to a new location
(c)
1. CN requests the location of MN
4. CN updates DNS and CN
5. CN reaches MN's new location
2. CN directly reaches the location of MN
CN
DNS
MN
MN
3. MN moves to a new location

Figure 15.1 Illustration of (a) Mobile IPv6, (b) distributed mobility management, and (c) identifier/locator split designs.

it frees MNs from mobility signaling and employs Mobile Access Gateways (MAG) to perform mobility management functions on behalf of MNs.

The major drawback of MIP and its extensions mentioned earlier is that all the packets from CN to MN have to take a detour to pass the HA, which is known as the triangle routing problem. Triangle routing can result in routing path stretch, which means the actual routing path is longer than the shortest one, as well as heavy load on HA. In recent years, a series of MIP derivatives [15–18], which follow Distributed Mobility Management (DMM) architectural paradigm [8, 33, 34], arise to address the problem. As shown in Figure 15.1b, DMM solutions distribute the functionality of HA to multiple mobility anchors deployed in the network so that the MN can always choose a nearby mobility anchor to maintain its binding cache and perform packet redirection. Thus, the MN's HoA never represents a fixed location, and triangle routing can be alleviated or even eliminated. In order to reach the MN, the relationship of the MN and its current mobility anchor is propagated among the deployed mobility anchors in the network, which can be realized in either a push or a pull mechanism [8, 33]. DMM research is still in the early stage, but it is considered as a promising way to evolve MIP networks and is currently under standardization in the IETF DMM group.

15.2.1.2 ILS Designs

ILS designs in the broad sense can be divided into two categories: one proposes to separate IP address space of core networks from edge networks, which is usually called core-edge separation. The main goal of core-edge separation protocols is to improve global routing scalability; thus, they usually focus on the network side. The other proposes a more clear separation of IP address's dual roles, and it usually introduces a new namespace as identifiers for the hosts/nodes in the Internet and treats the entire IP address space as locators. What is concerned in this chapter is the latter category, which always takes mobility handling as one of its goals. Host Identity Protocol (HIP) [19], Identifier/Locator Network Protocol (ILNP) [20], Name-Based Sockets (NBS) [21], and LISP Mobile Node (LISP-MN) [22] are typical solutions that fall into this category.

Compared with MIP that places identifier-to-locator mapping functions at the network side, ILS mobility solutions can be regarded as host-based solutions, since most mobility management functions are implemented at the host side. Figure 15.1c shows how these solutions works: when CN starts communication with MN, it first obtains the current IP address of the MN by querying a global mapping system (DNS plays the role in most cases), which always stores up-to-date identifier-to-locator mapping of each MN. When the MN moves to a new network, it keeps its identifier unchanged and obtains a new IP address as locator, and then the MN sends its new IP address to not only the global mapping system but also the CN side so that CN can directly reach its current location in time. Therefore, the mobility handoff is actually realized in an end-to-end way in such solutions. To keep session survivability, the transport layer only deals with identifiers in these solutions, and a mapping function is called to map the identifiers to IP addresses before data packets are sent out from the network layer. To realize such a function, these solutions either introduce a new layer or modify existing layers in the TCP/IP stack.

Though ILS protocols share the same core idea, they differ in ways to realize the idea including the formatting of identifiers, implementation of mapping functions on the host side and in the global mapping system, etc. HIP uses self-authenticating identifiers, called Host Identity (HI), to identify mobile hosts. HI is obtained by hashing the public key of a key pair that belongs to the host. With self-authenticating identifiers, each host is able to prove ownership to its HI through cryptographic methods. HIP use HI together with a port number to uniquely identify a transport layer session. To handle the mapping between HI and IP addresses on the host side, HIP inserts a new layer, called Host Identity Layer, between the network and transport layer in the protocol stack. HIP utilizes DNS together with additional rendezvous points to form a global mapping system, which stores the HI-to-IP address mappings of all mobile hosts. When communication begins, the initiator sends DNS request and fetches correspondent's HI and location of related rendezvous point. Then, the first data packet goes via the rendezvous point to reach the correspondent. After the initiator receives a data reply from the correspondent, following data stream travels directly between both communicating ends.

ILNP does not introduce new namespaces but utilizes IPv6 address space to identify mobile hosts, whose idea is derived from earlier research related to ILS [35]. ILNP splits IPv6 address space into an identifier part and a locator part: the first 64 bits of IPv6 address remains to be used for routing in the network, while the last 64 bits are used to uniquely identify mobile hosts. ILNP modifies the transport layer of mobile hosts to ensure that the session state only contains the identifier part of the entire IPv6 address (with port number). Different from HIP, ILNP completely relies on DNS to store the identifier-to-locator mappings.

NBS proposes to use domain names as identifiers of mobile hosts. NBS adopts a different approach to realize the mapping function by inserting a layer above the transport layer, which gives applications new socket interfaces and calls existing interfaces of TCP and UDP for data delivery. Using this method, mobility is hidden from the applications, and no change to TCP/IP stack is required. However, applications need to be redesigned to adapt the new socket interface, which implies that stale application cannot benefit from the mobility features offered by NBS. NBS also makes DNS as its global mapping system.

LISP-MN is developed based on research on Locator Identifier Separation Protocol (LISP) [36], which is a core-edge separation design. LISP proposes a separation of endpoint identifiers (EID) from routing locators (RLOC) and deploys ingress/egress tunnel routers (TR) to maintain EID-to-RLOC mappings. LISP-MN utilizes EID as identifier and RLOC as locators of mobile hosts and implements a lightweight TR on each mobile host to realize the mapping functions. LISP-MN does not rely on DNS, but makes use of several alternative map servers proposed by LISP as its global mapping system.

15.2.2 Problem Statement

Although there exist various methods to support Internet mobility, they have drawbacks in different aspects including triangle routing, large handoff latency, heavy signaling overhead, etc. This subsection gives a problem statement of existing Internet mobility solutions by analyzing mobility management functions of current solutions and pointing out a trade-off between routing path stretch and handoff efficiency.

15.2.2.1 Mobility Management Analysis

One of the main differences among existing Internet mobility solutions is how they implement handoff management functions. Handoff management is responsible for maintaining session survivability when the communicating nodes are moving. Different handoff management approaches can be classified into three categories. MIP comprises the first category, which we call local-scope handoff management, since the handoff signaling is always confined within a limited scope, that is, between the MN and the HA (or a local HA in HMIPv6 and PMIPv6). Thus, CNs only know one way to reach the MN, that is, via the HA. On the contrary, ILS designs adopt global-scope handoff management. The MN always sends mapping updates to the CN side, making all the CNs know the MN's exact location and send packets directly to the MN.

DMM solutions fall into the third category, which is a hybrid of the first two approaches: local-scope handoff signaling is always triggered to propagate the MN's mapping to a close HA, but global-scope handoff signaling is also required when the MN leaves one HA and attaches to another. Therefore, packets from the CNs can always be forwarded to an intermediary node that is close to the MN and knows how to reach the MN. Then the packets are routed to the MN based on its current IP address. Note that usually some agent near the CN is responsible for handling the signaling on behalf of the CN, making the handoff process transparent to CN.

There exist similarities in all three approaches. During handoff management, the MN must announce its up-to-date mappings into the network so that the CNs can reach the MN directly or indirectly. Specifically, some identifier-aware nodes in the network, we call rendezvous, must receive the mapping announcement and store the mappings, and then the CNs are able to reach the MN via the rendezvous (note that CNs themselves can also be rendezvous). The difference among these approaches is the scope of the mapping announcements: local scope, global scope, or a mixture of the two.

15.2.2.2 Routing Path Stretch and Handoff Efficiency

Existing efforts to handle Internet mobility expose a trade-off between routing path stretch and handoff efficiency: if the scope of mapping announcement is limited, as the MIP approach does, CNs may have to take a detour to reach the MN, which then causes routing path stretch; while if the mappings are announced to the CN side, as the ILS approach does, it may bring heavy overhead and large latency during the handoff, because the CNs can be distance from the MN and the number of CNs can be large.

A simple explanation of the trade-off is given here by drawing an analogy to the Internet routing. One common understanding in the routing research area is a fundamental trade-off between the routing table size and routing path stretch in a static network [37–39]. This trade-off implies a node in the network must store one routing table entry for each of the other nodes in the worst case to achieve the shortest path routing. Otherwise, one can only trade off an increase of the routing path stretch for a drop of the routing table size, because once a node loses the routing table entry for some remote node, it may not be able to forward packets to that node via the optimal path. The situation in mobility management is analogous: to ensure optimal routing path, mapping announcements must reach the CN side in the worst case; while if the scope of mapping announcement is limited to reduce the signaling overhead and latency, CNs will lose the exact location of the MN and have to reach the MN via indirection, which may lead to potential routing path stretch.

Internet mobility solutions take different ways to make the trade-off and get their own pros and cons: MIP only announces mappings to the HA; thus, it gains potential large routing path stretch but low handoff latency and overhead especially when MIPv6 extension (e.g., HMIPv6) is applied; ILS approach announces mappings to all the CNs; thus, it always gets none routing path stretch but may suffer from large handoff latency and overhead; DMM solutions are seeking a balance between the two.

It is still an open question on how to make the trade-off in DMM solutions. If an MN moves slowly and seldom changes HA, deploying HAs close to MNs can indeed reduce handoff latency and overhead without bringing routing path stretch. However, it may not be the case in real mobility scenarios. Considering the scenario that an MN simultaneously connects to multiple ISPs (e.g., the MN have both Wi-Fi and 3G/4G accesses), switch between different ISPs may become more common. Furthermore, a mobile user in future Internet may be able to switch ongoing communications from one device to another, which may also lead to frequent switch among ISPs. In both scenarios described earlier, there exists a large possibility that an MN changes HA frequently, which may significantly decrease the handoff efficiency of the hybrid handoff management. Therefore, it still needs investigation on how the third approach behaves when applied to more complex mobility patterns.

15.2.3 Mobility Management Based on SDN

To address the problems in current Internet mobility solutions, this chapter proposes to use SDN and OpenFlow. Before introducing the detailed protocol design, this subsection reviews existing research on SDN-based mobility and discusses benefits of handling Internet mobility using SDN.

15.2.3.1 Existing Research on SDN-Based Mobility

Researchers have already begun studying on how to offer better mobility support under SDN architecture. Yap et al. [40, 41] proposed OpenRoads to improve robustness during mobility handoff using multicast in OpenFlow networks. They showed how this is achieved by demonstration and also described their testbed deployment. In the following paper [42], they further abstract their idea as separating wireless services from infrastructures and rename OpenRoads to OpenFlow Wireless, which serves as a blueprint for an open wireless network. The focus in this chapter differs from the research earlier: this chapter focuses on improving basic IP mobility functions commonly adopted by existing protocols, while they paid more attention to adding new features, such as multicast, to basic mobility functions.

Pupatwibul et al. [43] proposed to enhance MIP networks using OpenFlow, which share similar goals to the proposal in this chapter. However, as will be shown, they only proposed one possible way to solve the problem, which may not be optimal in many scenarios, while this chapter abstracts the problem and gives a general discussion to seek for the best solution.

15.2.3.2 Benefits of SDN-Based Mobility

A summary of benefits of SDN-based mobility solution is already given in Section 15.1.2, while this subsection presents a further analysis based on the problem statement earlier. Recall the conclusion in the problem statement in Section 15.2.2.2, existing Internet mobility solutions adopt different ways to realize mapping announcement in handoff management and thus make different trade-offs between routing path stretch and handoff efficiency. SDN-based mobility solution can serve as a promising way to seek an ideal balance for the performance trade-off. It is because programmable devices and centralized control enable the flexibility to perform mapping announcement according to the MN's movement details. Specifically, since each device is programmable in SDN, they are all potential rendezvous for MNs, which makes SDN-based mobility management no longer restricted to local-scope or global-scope handoff management, but can perform mapping announcement in an arbitrary scope. Moreover, centralized control can decide in which scope the mappings should be announced according to the movement of MNs: if an MN moves within a limited area, only local-scope mapping announcement is sufficient, and along with the increasing of the MN's movement distance, the controller may find it necessary to announce the mappings to a larger scope.

Therefore, to seek an algorithm that optimizes the scope of mapping announcement in different mobility scenarios is one of the most important goals of this chapter. Section 15.3.3 describes how to seek such an algorithm, and it also proves optimality of the algorithm in terms of both optimal routing path and minimum handoff latency as well as signaling overhead.

15.3 Software Defined Internet Mobility Management

This section describes an SDN-based mobility management architecture for mobile Internet. Firstly, an overview of the architecture is given. Secondly, an instantiation of the architecture is presented that is designed using OpenFlow. Thirdly, an algorithm problem is addressed that serves as a key component of the architecture. At last, an implementation of the architecture is presented together with experiments to compare the proposal with existing Internet mobility solutions.

15.3.1 Architecture Overview

Internet mobility management functions based on SDN can be separated into control plane functions and data plane functions as demonstrated in Figure 15.2. In the control plane, two subfunctions are required to realize mobility management. One subfunction requires SDN controllers to collect the current location of each MN and maintain a mapping for each MN. The mapping dynamically binds identifier of an MN to locator of the MN. The definition of identifier is the same as that in ILS-related researches, that is, some stable information that does not need to change when the MN changes its location in the network. Identifiers do not have a restricted format but can be any field in the packet that can be recognized by SDN controllers and devices. MN's locators should be represented by some packet field that can be used to reach the MN's current location. Normally, IP addresses serve as locators of MNs.

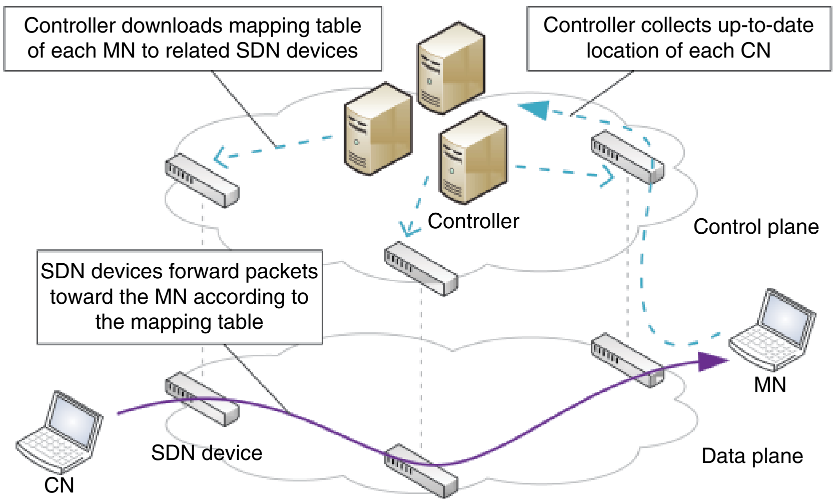
Controller downloads mapping table of each MN to related SDN devices
Controller collects up-to-date location of each CN
Controller
Control plane
SDN devices forward packets toward the MN according to the mapping table
MN
CN
SDN device
Data plane

Figure 15.2 Architecture overview.

To realize this subfunction, SDN devices are required to inform controllers of the attachment and detachment of each MN. When an MN leaves one SDN domain and enters another, inter-SDN domain mechanism is required to synchronize mappings of the MN between controllers in different domains.

The other control plane subfunction requires the controller to download each MN's mapping to related SDN devices. It can be subdivided into two cases: in the first case, the controller downloads an MN's mapping to the SDN devices that are requesting the mapping; in the second case, after the controller updates an MN's mapping, it downloads the mapping to some SDN devices that has already stored the mapping. The purpose of the downloading in the second case is to replace stale mappings on SDN devices with up-to-date ones so that packets toward the MN can be forwarded to its current location in time. Note that both CN and MN are hardly involved in the control plane functions and most mobility management functions in the control plane are realized on the network side.

In the data plane, SDN devices directly receive packets destined to MNs from CNs and forward the packets according to the mappings downloaded from controllers. When an SDN device lacks required mapping for packet delivery, it triggers a control plane function to request the mapping from the controller.

The control and data plane functions described above comprise the basic mobility management functions based on SDN. Protocol details such as how the mappings are collected and downloaded are explained using an OpenFlow-based example in the following subsections.

15.3.2 An OpenFlow-Based Instantiation

This subsection describes an OpenFlow-based instantiation of the proposed architecture. Note that though the detailed protocol design described in this chapter is based on OpenFlow, the proposed architecture can also be designed and implemented in a similar way using other techniques that realize the idea of SDN.

15.3.2.1 Protocol Description

The OpenFlow-based design employs IP addresses to identify and locate MNs. Like all the other mobility protocols, a stable identifier is assigned to each MN. The identifier is also called HoA, which is nonroutable and should belong to a specific address block. An MN's location is represented by CoA, which is not owned by MN, but its first-hop OpenFlow switch. It means MNs never require to reconfigure IP addresses when attaching to new networks, but the network side helps to accomplish the work, which is similar to PMIPv6. CoAs are routable addresses and thus are used to reach MNs when they are moving around.

OpenFlow controller is responsible for maintaining binding cache that maps an MN's HoA to CoA. For each MN, a subset of OpenFlow switches in the network serve as indirection point for the MN. They store replica of the MN's binding cache in the form of flow table, which is downloaded from the controller, and redirect packets toward the MN according to the flow table.

Figure 15.3 illustrates how CN reaches MN in both communication initiation and handoff procedures. HoA of MN and CN are IP_M and IP_C, respectively. First, when switch S3 detects the attachment of MN, it learns the MN's HoA, assigns a CoA IP_S3 to the MN, and then sends a Binding Update message, which contains a (IP_M, IP_S3) tuple to its controller. The controller stores the binding locally and immediately downloads a flow table entry to S3 that indicates "for all packets with destination address IP_S3, rewrite their destination addresses to IP_M."

The communication initiation process is described as follows, assuming that CN is communicating with MN: since CN only knows HoA of the MN, the destination address in the packets it sends to MN is IP_M. When CN's first-hop switch S1 receives such a packet, it learns IP_M is nonroutable, and there are no local flow tables that match the address. Thus, it forwards the packet to its controller via packet-in. The controller (for simplicity, here, the controllers of S1 and S3 are assumed to be the same one, and the case with multiple controllers will be discussed in the following subsection) looks IP_M up in local binding cache table and gets the corresponding CoA. Then the controller forwards the packet to S3 via packet-out and at the same time places the binding cache to S1 by downloading a flow table entry indicating "for all packets with destination address IP_M, rewrite their destination addresses to IP_S3." Then the following packets can flow directly from CN to MN: first from S1 to S3 and then from S3 to the MN.

The handoff process is described as follows, assuming that the MN leaves S3 and attaches to S4: similarly, detecting the attachment, S4 assigns IP_S4 to the MN and sends Binding Update to the controller. The controller receives the update and learns that the MN has just moved; thus, it is responsible for accomplishing the handoff by modifying existing CN-to-MN flow path toward MN's new location. Take the scenario in Figure 15.3 as an example: since the controller knows how the flow goes from CN to MN, it places the new binding cache to S2

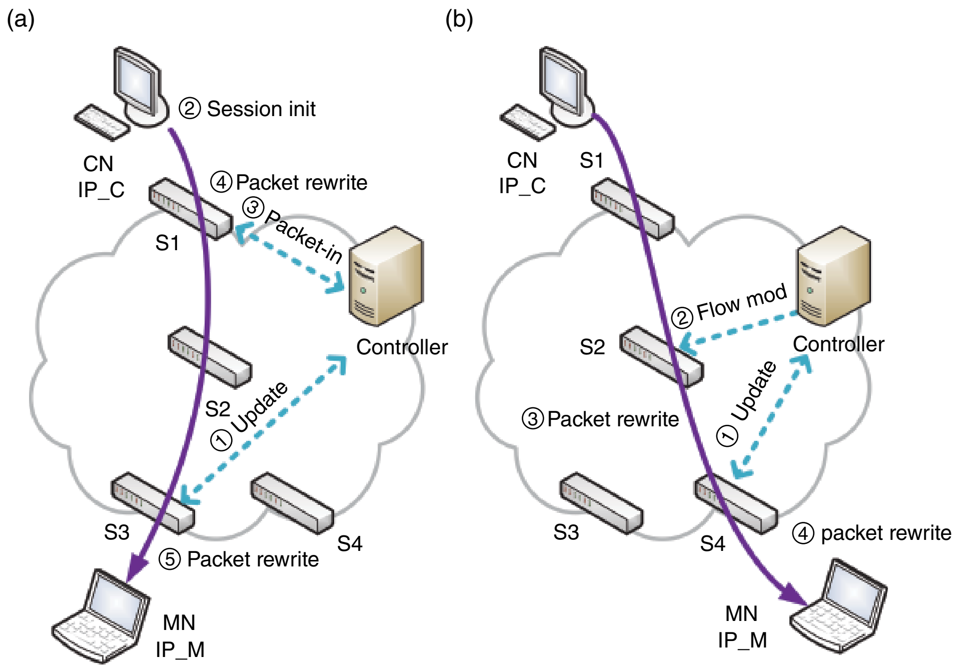

Figure 15.3 (a) Communication initiation and (b) handoff processes.

by downloading a flow table entry to S2 that indicates "for all packets with destination address IP_S3, rewrite their destination addresses to IP_S4." Then the new CN-to-MN flow will go through three redirections: S1 to S2, S2 to S4, and S4 to MN. In practice, there may exist various ways to place the binding cache, and Figure 15.3 only shows one possibility. The binding cache placement algorithm is further discussed in Section 15.3.3.

15.3.2.2 Discussions

If CN and MN are located far apart from each other or located in different domains, it is possible that the controllers of their first-hop switches are different. If the two controllers belong to the same administrative domain, the problem may become simpler since intradomain communication between controllers is more common. If the two controllers belong to different administrative domains, interdomain interactions between controllers are required, which may bring larger cost comparing with the intradomain case. Specifically, interdomain communication initiation between MN and CN is less costly than interdomain handoff of MN, because the former only requires a query and response of binding cache and is easier to handle, but the latter requires the controller to know the CN-to-MN flow path, which is not a preknowledge of the controller in interdomain case.

However, interdomain handoff is not common in practice. There are two common ways to trigger an interdomain handoff: in one case, the MN moves for a long distance and then leaves one domain and enters another, which can be quite infrequent; in the other case, the MN switches between different providers (e.g., different Wi-Fi or 3G/4G networks) without long-distance movement. As for the second case, its occurrence probability can be further reduced by making multiple heterogeneous local networks be controlled by one logical controller. Using this method, MN's switching among different access networks is analogous to intradomain handoff.

However, though infrequent, interdomain handover is unavoidable. To improve interdomain handover efficiency, the protocol can temporarily fall back to triangle routing that only requires one flow table downloading to the MN's previously attached switch. After MN–CN communication is restored, further operations can be performed to optimize the path between MN and CN.

15.3.3 Binding Cache Placement Algorithm

This subsection further researches into the binding cache placement during MN's handoff procedure. Theoretically, any switch on the CN-to-MN flow path before MN's movement (e.g., S1, S2, and S3 in Fig. 15.3) can serve as a candidate switch, which is called Target Switch (TS). However, choosing some TS may lead to serious performance drawbacks. For example, it is a straightforward idea to choose MN's first-hop switch before movement (e.g., S3 in Fig. 15.3) as TS, but this method will result in triangle routing in most cases. Another idea is to choose CN's first-hop switch (e.g., S1 in Fig. 15.3) as TS, but this method may result in a large number of flow table downloading and high handoff latency, which is analogous to the end-to-end Binding Update manner adopted by HIP-like protocols. Therefore, the binding cache placement problem (BCPP) requires further study. The following of this section formalizes and solves the problem.

15.3.3.1 BCPP

First, the goals of the binding cache placement algorithm are given as follows:

Goal 1: Keep optimal forwarding path. This goal ensures the shortest forwarding data path between MN and CN and avoids triangle routing.
Goal 2: Minimize the distance between MN and TS. The purpose of proposing this goal is to localize the signaling caused by MN's mobility events.
Goal 3: Minimize flow entry downloading per movement. This goal can help to both limit the mobility-related flow table maintained on switches and reduce the signaling overhead introduced by flow table downloading.

Then a general *BCPP* is defined as an optimization problem: given a set of TS to place the binding cache for an MN, the BCPP is to find a subset of the switches that optimizes some goals.

However, the proposed goals conflict with each other in many cases, for example, selecting MN's first-hop switch before movement as TS will always satisfy Goal 3 but has a large possibility to conflict with Goal 1. Thus, BCPP is further specified into the following two problems:

BCPP-1: BCPP that takes Goal 2 as optimization objective and Goal 1 as constraint.
BCPP-2: BCPP that takes Goal 3 as optimization objective and Goal 1 as constraint.

15.3.3.2 Problem Formalization and Solution

BCPP-1

Assume that during a handoff procedure, MN moves from switch s_n to $s_{n'}$ and CN stays attaching to switch s_1 as shown in Figure 15.4. A set of definitions are given before formalizing BCPP-1:

Definition 15.1

$\text{path}_{\text{prev}}$ is defined as a set of switches on the MN–CN path before movement of MN, for example, $\{s_1, s_2, \ldots, s_i, \ldots, s_n\}$ in Figure 15.4.

$\text{path}_{\text{current}}$ is defined as a set of switches on the MN–CN path after movement of MN, for example, $\{s_1, s_2, \ldots, s_i, \ldots, s_{n'}\}$ in Figure 15.4.

Path pair is a ($\text{path}_{\text{prev}}$, $\text{path}_{\text{current}}$) tuple.

Switch *s* satisfies path pair *p* means after placing the binding cache (of the MN) on s, the new MN–CN path path_{new} equals to $\text{path}_{\text{current}}$. This ensures Goal 1, that is, optimality of the forwarding path (no triangle routing).

Then BCPP-1 is formalized as:

Problem 15.1

Given each path pair p, find a switch s that satisfies p and at the same time minimizes its distance to the MN.

The solution to Problem 15.1 is relatively simple. First, another group of definitions is given as follows:

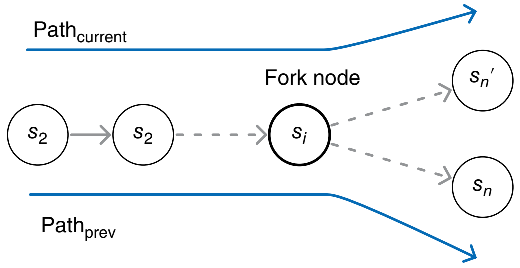

Figure 15.4 This figure helps to demonstrate Definitions 15.1, 15.2, and 15.3: path$_{\text{prev}}$ consists of nodes from s_1 to s_n, while path$_{\text{current}}$ consists of nodes from s_1 to $s_{n'}$, and s_i is the fork node.

Definition 15.2
Satisfactory switch set C_p for path pair p is defined as $\forall s \in C_p$, s satisfies p, for example, $\{s_1, s_2, \ldots, s_i\}$ is a satisfactory switch set for path pair (path$_{\text{prev}}$, path$_{\text{current}}$) in Figure 15.4.

Fork node of path pair p is defined as the node where two paths of the path pair "fork," for example, s_i is the fork node of path pair (path$_{\text{prev}}$, path$_{\text{current}}$) in Figure 15.4.

Then the solution to Problem 15.1 is given:

Algorithm 15.1
Given a path pair p, find its fork node s.

The complexity of the algorithm is $O(d \cdot n)$ where n represents number of path pairs and d represents length of the path. Related proof is omitted here.

BCPP-2
More definitions are given to formalize BCPP-2:

Definition 15.3
Switch s satisfies a set of path pairs P means $\forall p \in P$, s satisfies p.
A set of switches S satisfies a set of path pairs P means $\forall p \in P$, $\exists s \in S$ s.t. s satisfies p.

Problem 15.2
Given a set of n path pairs P, find the smallest set of switches S that satisfies P.

Problem 15.2 can be solved in two steps:

Step 1: For each path pair p, find the largest satisfactory switch set C_p.
Step 2: Find the smallest set S s.t. for each C_p, $S \cap C_p \neq \emptyset$.

The complexity of Step 1 is $O(d \cdot n)$. Step 2 can be reduced by *Set Covering Problem* that is NP hard; thus, Problem 15.2 is an NP-hard problem.

Obviously, Problem 15.2 can be solved using exhaustive search, but its complexity is $O(d^n)$ and is unacceptable. We find that under certain circumstance, Problem 15.2 can be solved using a simple algorithm. The circumstance is described as:

Assumption 15.1
Two paths to the same destination share identical "suffix" after they "meet."

Assumption 15.1 is satisfied as long as packet forwarding between MN and CN only relies on destination IP address, which is a common case in current intradomain scenarios. With this assumption, the solution to Problem 15.2 is proposed as the following algorithm:

Algorithm 15.2
Find the fork node s_i of each path pair $p \in P,\ S = \cup\{s_i\}$.

Actually, Algorithm 15.2 is similar to Algorithm 15.1 except that Algorithm 15.2 works on a set of path pairs. Algorithm 15.2 takes $O(d \cdot n)$ to get the optimal result. The proof of the optimality of Algorithm 15.2 is omitted in this chapter.

Obviously, Algorithm 15.2 also optimizes Problem 15.1. Thus, if Assumption 15.1 is satisfied, Algorithm 15.2 can generate optimal results for both Problems 15.1 and 15.2, which means all three goals can be simultaneously achieved.

When Assumption 15.1 cannot be satisfied, another algorithm is given to solve Problem 15.2:

Algorithm 15.3
Greedy Set Covering.

Step 1: Let $X = P,\ S = \varnothing$.
Step 2: Repeat the following process until $X = \varnothing$; find *i* s.t. S_i contains the largest number of elements in *X*, and then let $S = S \cup \{s_i\},\ X = X \setminus S_i$.

According to existing research [44], Greedy Set Covering algorithm takes $O(d \cdot n^2)$ to get a result with approximation ratio $\ln\ n + 1$.

Note that Algorithm 15.3 also generates optimal result when Assumption 15.1 is satisfied. The proof is omitted in this chapter.

15.3.3.3 Evaluation

This subsection evaluates the previously proposed algorithms to see how they perform in real network topologies. Since it is difficult to get real intradomain routing data that conflicts with Assumption 15.1, only Algorithm 15.2 is evaluated under Assumption 15.1 using real intradomain topology with the shortest path routing. Two additional algorithms are introduced as comparatives:

Algorithm-random: For each path pair *p*, this algorithm randomly selects a switch *s* that satisfies *p* as TS.

Algorithm-CN: For each path pair *p*, this algorithm selects the first-hop switch of CN as TS.

All three algorithms satisfy Goal 1; thus, the comparison uses metrics from the other two goals: one metric is MN–TS distance, and the other metric is the number of binding cache downloaded per MN per movement.

The evaluation topology and routing data are calculated using intradomain topologies from RocketFuel [45] including AS1221, AS1755, AS6461, AS3257, AS3967, and AS1239. As evaluations based on different topologies show similar results, three of them are chosen to demonstrate the results, and they are AS1221 with 208 nodes, AS3257 with 322 nodes, and AS6461 with 276 nodes. To study the performance of the algorithm based on various topologies, another two topologies are generated to add differentiation: one is a 200-node hierarchical topology with a densely interconnected core network and several treelike edge networks, and the other is a 200-node flat topology in which nodes randomly connect to each other with an average degree.

For each one of the topologies above, the evaluation runs for 100 turns. In each turn, a different node in the topology is selected as the MN, and 10 randomly located nodes are chosen as CNs. The MN performs 10 movements per turn using a modified Markov chain-based random walk model: during each movement, the MN randomly attaches to a new node that is one hop away from its previous location.

Evaluation results are demonstrated in Figure 15.5. As shown in the figures, Algorithm 15.2 has the lowest value. Figure 15.5a shows that MN–TS distance of Algorithm 15.2 only takes 10–20% of the network diameter, which means TS is located about two hops away from MN on average, and this offers a good guarantee on the handoff efficiency. Algorithm-CN has the largest MN–TS value since it always pushes binding cache to the CN side. The value of Algorithm-random stays between Algorithm-CN and Algorithm 15.2. Also, when evaluation topology becomes more flat, MN–TS values of three algorithms approximate to each other. It is because with the "flatten" of topology, average distance between nodes also drops.

Figure 15.5b shows that, on average value, Algorithm 15.2 only generates about 0.3–0.5 flow table downloading per each CN in three intradomain topologies, while the other two algorithms always require downloading one flow table for each CN. Again, Algorithm 15.2 requires more flow table downloading in flat topologies. Just as the fact that hierarchical topology helps to reduce routing table size, it also helps to reduce binding cache maintenance.

15.3.4 System Design

This subsection describes system design of the proposal including an implementation based on Mininet and experiments that compares this proposal with another two representative Internet mobility protocols.

15.3.4.1 Implementation

The protocol is implemented based on Mininet 2.1.0 [46]. Pox [47] is chosen as the controller in the implementation. Figure 15.6a demonstrates the protocol flow. When switch S3 detects MN's attachment, it assigns a CoA to the MN and registers (HoA, CoA) tuple on the controller using port-status message. Receiving registration from S3, controller stores the binding cache locally and downloads a "rewrite" flow entry (the same as that in Section 15.3.2.1) to S3 using flow-mod message.

When switch S1 receives a packet toward an unknown host, it sends the packet to controller using packet-in message. Upon receiving packet-in, controller rewrites its destination IP address and resends it out using packet-out message. At the same time, controller downloads MN's binding cache to S1 using flow-mod message. In order to generate the path pairs used in the binding cache placement algorithm, the controller needs to keep a record of all the CN-to-MN paths, which we call *Path Record* (*PR*). For example, in this case, when the controller downloads MN's binding cache to S1, it adds one entry into PR indicating that "There exists a flow from S1 to the MN." When the binding cache on S1 expires, S1 will acknowledge the controller, and then the controller will delete the related entry in PR.

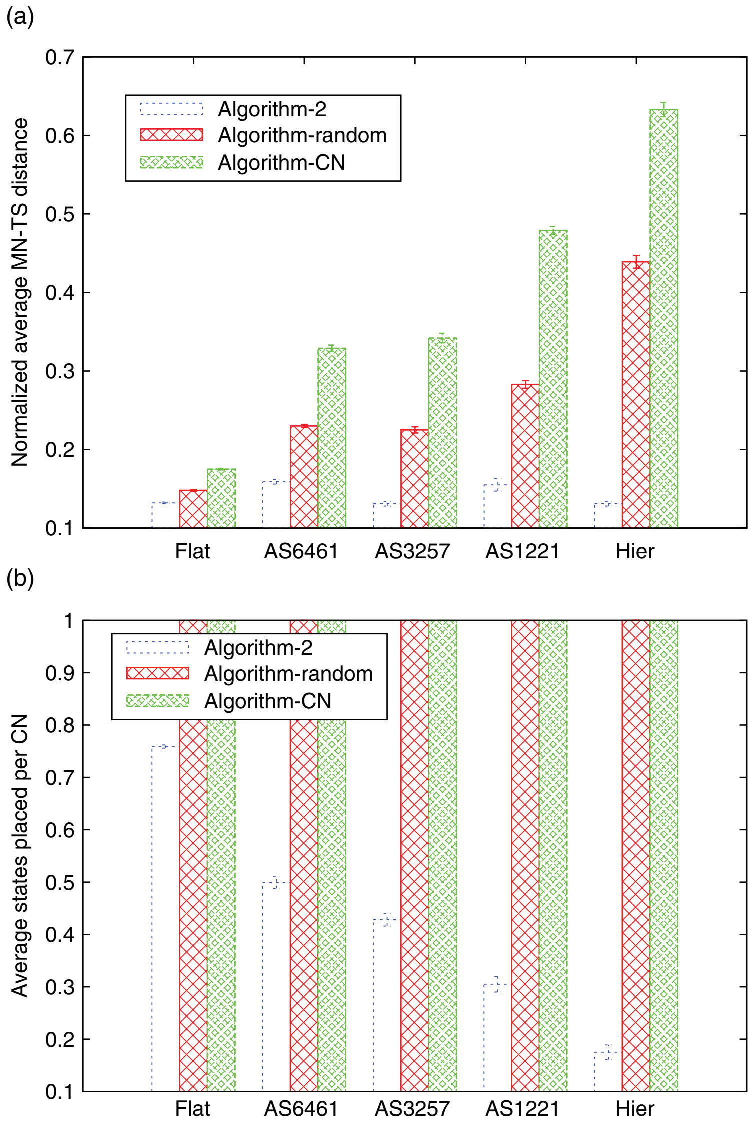

Figure 15.5 The two figures show results of comparison among three proposed algorithms based on five different topologies. (a) Shows normalized average MN–TS distance, and (b) shows average number of binding cache placed per CN. Algorithm 15.2 outperforms the other two in all scenarios.

After MN moves to S2, another similar procedure handles related registration and flow table downloading procedures. To deal with the handoff, the controller runs the binding cache placement algorithm discussed previously. It uses PR to get all path pairs related to the MN (only one path pair in this case). To obtain the path pairs, the controller looks up in the

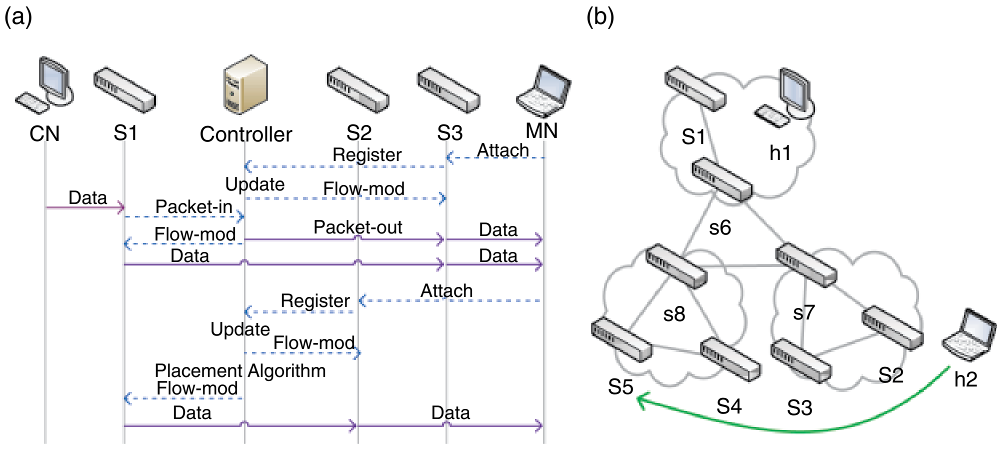

Figure 15.6 (a) Protocol flow and (b) experiment topology of the implementation.

PR and calculates the previous path from S1 to MN via S3, as well as the current path from S1 to MN via S2. After running the binding cache placement algorithm, the controller gets a set of switches that require updating. Then the controller downloads the MN's up-to-date binding cache to these switches using flow-mod messages. Note that the controller also needs to update PR after this turn of flow table downloading.

15.3.4.2 Experiment

Methodology

Several experiments are made based on the implementation to compare the proposed protocol with another two IP mobility protocols: PMIPv6 and ILNP. The two protocols are chosen because they serve as good representatives of the solutions reviewed in Section 15.2.1: a network-based protocol and a host-based protocol. To make comparisons, another two controllers are implemented to realize the basic mobility functions of PMIPv6 and ILNP, respectively. PMIPv6 is easier to implement based on Mininet since it is a network-based protocol. But ILNP is more difficult; thus, the protocol is simulated in an approximate way: the mobility functions are moved from hosts to their first-hop switches.

The experiment topology is shown in Figure 15.6b, which consists of one controller, two hosts, and eight switches. The topology is divided into three interconnected subdomains: (S7, S2, S3), (S8, S4, S5), and (S6, S1). Delays of intersubdomain links, intrasubdomain links, and "wireless links" (between H2 and attached switches) are 20, 2, and 10 ms, respectively. Bandwidths of the above three types of links are 100, 100, and 10 Mbps, respectively. Since in the current version of Mininet in-band control between switches and controller is not supported, thus, control traffic is out of band in the experiments. H2 serves as the MN and moves back and forth between switches S2 and S5. H1 serves as the CN and keeps immobile. The experiments run Iperf, which is a commonly used network testing tool, between the two hosts and collect end-to-end performance including round-trip time (RTT), packet loss rate, as well as throughput.

When simulating PMIPv6 in this topology, S7 serves as the HA; S2, S3, S4, and S5 serve as MAG; and S7 and S8 serve as Local Mobility Anchor (LMA). H2's moving from S3 to S4 indicates that it leaves its home network and needs to rely on S7 and S8 for packet indirection. When simulating ILNP, each time H2 moves, its first-hop switch will send Binding Update to S1 on behalf of H2, and then S1 handles the update on behalf of H1. Note that the experiments actually favor PMIPv6 and ILNP for two reasons: firstly, IP reconfiguration is ignored in the

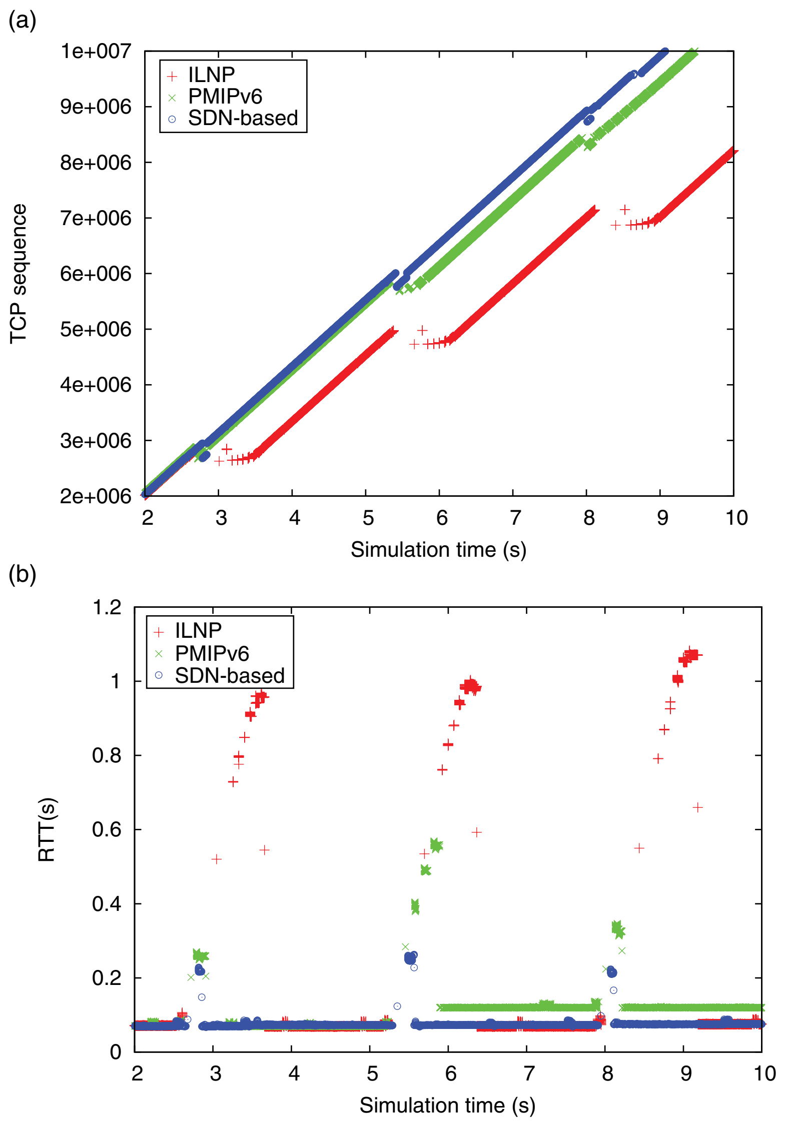

Figure 15.7 (a) TCP sequence and (b) round-trip time (RTT) of three simulated protocols during three handoff events in 10 s.

handoff process of the two protocols. Secondly, Binding Update process is simplified and only takes one-way delay: MN-to-HA delay in PMIPv6 case and MN-to-CN delay in ILNP case. Both simplifications help to improve handoff efficiency of the two protocols.

Experiment Results

The first experiment runs Iperf between H1 and H2 for 10 s during which period H2 moves from S2 to S5 and performs three handoffs. Figure 15.7a shows collected TCP sequence of PMIPv6, ILNP, and the SDN-based solution within the simulation time, from which we can infer that SDN-based solution generates smoother handoff than the other two: TCP based on ILNP experiences time-out and slow start during each handoff, which makes ILNP performs the worst in the experiment. It is because ILNP needs to send Binding Update from the MN side toward the CN side, and this may seriously degrade handoff efficiency especially when both sides are located away from each other. TCP based on PMIPv6 experiences only one time-out during the second handoff, as the other two handoffs can be handled locally by LMA, while the second handoff is an intersubdomain handoff and requires interactions with HA. In contrast, TCP on the SDN-based solution can always recover from packet loss during handoff using fast retransmit.

Figure 15.7b shows RTT of three protocols collected in the same experiment scenario, where we observe that RTT value of all three protocols temporarily raises to a higher value during handoff process. Besides, RTT of PMIPv6 stays at a higher value after the second handoff. It is because when H2 leaves the home network (after moving from S3 to S4), all packets to H2 are relayed by the HA S7, which results in triangle routing. The SDN-based solution avoids triangle routing as the binding cache placement algorithm ensures optimal forwarding path, and in this scenario, it is achieved by downloading binding cache to S6.

15.4 Conclusion

This chapter addresses mobility in IP network under SDN architecture. SDN has advantages in handling problems in current mobility protocols because of its programmable devices, centralized control, as well as other features. By proposing an SDN-based mobility management architecture together with an OpenFlow-based protocol design, implementation, and experiments, this chapter demonstrated that SDN enables the flexibility of mobility management, making it adaptive to various mobility scenarios in future mobile Internet.

References

[1] L. Zhang, R. Wakikawa, and Z. Zhu. Support Mobility in the Global Internet. In Proceedings of the 1st ACM Workshop on Mobile Internet through Cellular Networks, ACM, 2009. Beijing, China.

[2] P. Zhang, A. Durresi, and L. Barolli. A Survey of Internet Mobility. In International Conference on Network-Based Information Systems 2009, NBIS'09, 147–154. IEEE, 2009. Indianapolis, USA.

[3] F. M. Chiussi, D. A. Khotimsky, and S. Krishnan. Mobility Management in Third-Generation All-IP Networks. Communications Magazine, IEEE, 2002, 40(9): 124–135.

[4] D. Saha, A. Mukherjee, I. S. Misra, and M. Chakraborty. Mobility Support in IP: A Survey of Related Protocols. Network, IEEE, 2004, 18(6): 34–40.

[5] I. F. Akyildiz, J. Xie, and S. Mohanty. A Survey of Mobility Management in Next-Generation All-IP-Based Wireless Systems. Wireless Communications, IEEE, 2004, 11(4): 16–28.

[6] A. Lucent. Introduction to Evolved Packet Core. Strategic White Paper, 2009. Available from www.alcatel-lucent.com (accessed February 19, 2015).

[7] O. Tipmongkolsilp, S. Zaghloul, and A. Jukan. The Evolution of Cellular Backhaul Technologies: Current Issues and Future Trends. Communications Surveys & Tutorials, IEEE, 2011, 13(1): 97–113.

[8] J. C. Zuniga, C. J. Bernardos, A. de la Oliva, T. Melia, R. Costa, and A. Reznik. Distributed Mobility Management: A Standards Landscape. Communications Magazine, IEEE, 2013, 51(3): 80–87.

[9] J. N. Chiappa. Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture. 1999, Available from http://mercury.lcs.mit.edu/~jnc/tech/endpoints.txt (accessed January 24, 2015).

[10] Z. Zhu, R. Wakikawa, and L. Zhang. A Survey of Mobility Support in the Internet. RFC 6301, IETF, 2011.

[11] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944, IETF, 2010.

[12] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275, IETF, 2011.

[13] H. Soliman, C. Castelluccia, K. ElMalki, and C. Castelluccia. Hierarchical Mobile IPv6 (HMIPv6) Mobility Management. RFC 5380, 2008.

[14] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213, IETF, 2008.

[15] R. Wakikawa, G. Valadon, and J. Murai. *Migrating Home Agents towards Internet-scale Mobility Deployment*. In Proceedings of the 2006 ACM CoNEXT Conference. ACM, 2006. Lisboa, Portugal.

[16] M. Fisher, F.U. Anderson, A. Kopsel, G. Schafer, and M. Schlager. A Distributed IP Mobility Approach for 3G SAE. In 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2008), IEEE, 2008. Cannes, French Riviera, France.

[17] R. Cuevas, C. Guerrero, A. Cuevas, M. Calderón, and C.J. Bemardos. P2P Based Architecture for Global Home Agent Dynamic Discovery in IP Mobility. In 65th IEEE Vehicular Technology Conference, IEEE, 2007. Dublin, Ireland.

[18] Y. Mao, B. Knutsson, H. Lu, and J. Smith. DHARMA: Distributed Home Agent for Robust Mobile Access. In Proceedings of the IEEE Infocom 2005 Conference, IEEE 2005. Miami, USA.

[19] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423, IETF, 2006.

[20] R. Atkinson and S. Bhatti. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740, IETF, 2012.

[21] J. Ubillos, M. Xu, Z. Ming, and C. Vogt. Name-Based Sockets Architecture. IETF Draft, 2011. Available from https://tools.ietf.org/html/draft-ubillos-name-based-sockets-03 (accessed February 19, 2015).

[22] D. Farinacci, D. Lewis, D. Meyer, and C. White. LISP Mobile Node. IETF Draft, 2013. Available from http://tools.ietf.org/html/draft-meyer-lisp-mn-09-12 (accessed February 19, 2015).

[23] National science foundation future internet architecture project. Available from www.nets-fia.net (accessed February 19, 2015).

[24] I. Seskar, K. Nagaraja, S. Nelson, D. Raychaudhuri. MobilityFirst Future Internet Architecture Project. In Proceedings of the 7th Asian Internet Engineering Conference. pp. 1–3. ACM. Bangkok, Thailand.

[25] A. Venkataramani, A. Sharma, X. Tie, H. Uppal, D. Westbrook, J. Kurose, and D. Raychaudhuri. Design Requirements of a Global Name Service for a Mobility-centric, Trustworthy Internetwork. In Fifth International Conference on Communication Systems and Networks (COMSNETS). pp. 1–9. IEEE, 2013. Bangalore, India.

[26] D. Han, A. Anand, F.R. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D.G. Andersen, J.W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient Support for Evolvable Internetworking. In Proceedings of the 9th USEnIX NSDI. ACM, 2012. San Jose, USA.

[27] L. Zhang, A. Afanasyev, and J. Burke. Named Data Networking. Technical Report, 2014. Available from http://named-data.net/publications/techreports (accessed February 19, 2015).

[28] Z. Zhu, A. Afanasyev, and L. Zhang. A New Perspective on Mobility Support. Technical report, 2013. Available from http://named-data.net/publications/techreports (accessed February 19, 2015).

[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review, vol. 38, pp. 69–74, 2008.

[30] J. Saltzer. On the Naming and Binding of Network Destinations. RFC 1498, IETF, 1993.

[31] E. Lear and R. Droms. What's in a Name: Thoughts from the NSRG. IETF Draft, 2003. Available from http://tools.ietf.org/html/draft-irtf-nsrg-report-10 (accessed February 19, 2015).

[32] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, IETF, 2007.

[33] H.A. Chan, H. Yokota, P.S.J. Xie, and D. Liu. Distributed and Dynamic Mobility Management in Mobile Internet: Current Approaches and Issues. Journal of Communications, 2011, 6(1): 4–15.

[34] IETF. Distributed Mobility Management (DMM). IETF Working Group. Available from http://tools.ietf.org/wg/dmm/ (accessed January 24, 2015).

[35] M. O'Dell. GSE—An Alternate Addressing Architecture for IPv6. IETF Draft, 1997. Available from http://tools.ietf.org/html/draft-ietf-ipngwg-gseaddr-00 (accessed February 19, 2015).

[36] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller. The Locator/ID Separation Protocol (LISP). RFC 6830, IETF, 2013.

[37] D. Peleg and E. Upfal. A Trade-off between Space and Efficiency for Routing Tables. In 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 43–52. ACM, 1988. Chicago, IL, USA.

[38] C. Gavoillz and S. Pérennès. Memory Requirement for Routing in Distributed Networks. In Proceedings of the 15th PODC. ACM, 1996. Philadelphia, PA, USA.

[39] D. Krioukov, K.C. Claffy, K. Fall, and A. Brady. On Compact Routing for the Internet. ACM Computer Communications Review, 2007, 37(3): 41–52.

[40] K. Yap, T.Y. Huang, M. Kobayashi, M. Chan, R. Sherwood, G. Parulkar, and N. McKwown Lossless Handover with n-Casting between WiFi-WiMAX on OpenRoads. In ACM Mobicom. ACM, 2009. Beijing, China.

[41] K. Yap, M. Kobayashi, D. Underhill, S. Seetharaman, P. Kazemian, and N. Mckwown. The Stanford Openroads Deployment. In ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH), ACM, 2009. Beijing, China.

[42] K. Yap, R. Sherwood, M. Kobayashi, T.Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar. Blueprint for Introducing Innovation into Wireless Mobile Networks. In Workshop on Virtualized Infrastructure Systems and Architectures, pp. 25–32. ACM 2010. New Delhi, India.

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

[44] V. Chvatal. A Greedy Heuristic for the Set-Covering Problem. Mathematics of Operations Research, 1979, 4(3): 233–235.

[45] Rocketfuel: An ISP Topology Mapping Engine. Available from www.cs.washington.edu/research/networking/rocketfuel/ (accessed January 24, 2015).

[46] Mininet: An Instant Virtual Network on Your Laptop (or other PC). Available from http://mininet.org/ (accessed January 24, 2015).

[47] POX Controller. Available from www.noxrepo.org/pox/about-pox/ (accessed January 24, 2015).