Seamless Handover between CDMA2000 and 802.11 WLAN using mSCTP

A thesis
Submitted in partial fulfilment
of the requirements for the degree
of

Master of Engineering
in
Electrical and Computer Engineering
by

Deng Feng

Supervisor
Professor Harsha R.Sirisena

Department of Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand

December 2006

Abstract

With the deployment of 3G networks and gradual implementation of wireless networks, seamless handover between these wireless networks is becoming an increasingly desirable. mSCTP (Mobile Stream Control Transmission Protocol) is a new protocol developed from SCTP (Stream Control Transmission Protocol) to provide seamless handover based on IP networks. This thesis studies how to use this new protocol to handle handovers on transport level between CDMA2000 and WLAN networks.

A survey of recently proposed and used mobility protocols is presented, comparing three common handover protocols operating on different layers: MIP (mobile IP) for the network layer, mSCTP for the transport layer and SIP (Session Initial Protocol) for the session layer. The results show mSCTP is the future for mobility support.

Lastly, I will present a detailed procedure on how to set up handover testbed between CDMA2000 network and 802.11 WLAN based on mSCTP and the results show that the handover performed between these two networks is fast and smooth but it is affected by the signal strength of the CDMA2000.

Acknowledgement

I would like to express my sincere thanks and gratitude to my supervisor, Professor Harsha Sirisena for his much guidance and support during the time of my research. My thanks to Professor Krzysztof Pawlikowski for guidance and help in building the testbed.

I would like to thank Telecom New Zealand for funding my research and providing the hardwares and other necessary equipment to build the testbed and without them it would not have been possible.

Special thanks to Lee Begg and Dave van Leeuwen for their help and advices in setting up and debugging my testbed. To my colleagues in the Network Research Group, I would like to thank you all for the times we spend together in sharing knowledge in all the meetings.

To my parents, thank you for the love, support and prayers throughout these years when I am studying in New Zealand. Thanks for the encouragement you give to me so I never give up and do my best to finish my thesis. Lastly, thank everyone who cares about me.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

The invention of mobile devices such as laptops, hand-held computers, and cellular phones has changed the whole Internet world. The main feature of mobile devices is that they can access the Internet wirelessly by different air interfaces using different technologies while roaming. This makes mobile devices become increasingly popular.

However, the traditional Internet does not support any needed features and architectural structures for mobility management. The traditional addressing scheme of the Internet has been designed based on the assumption that any node only has one fixed and permanent IP address in the Internet. Consequently, any node could be easily identified by this unique IP address and its location is directly recognized in the Internet.

When a mobile device comes to the Internet, it will move from one sub-network to another sub-network. However, the IP address of the mobile device will keep unchanged and the IP address can not reflect the new location due to the traditional addressing scheme of the Internet. As a result, the packets from the new location will not be routed to the destination and the packets to the new location of the mobile device will be forwarded to the old location by the traditional Internet protocols. All the connections for the mobile device will be lost during the movement and this is regarded as IP mobility management issues. [1].

IP mobility issues have been regarded as the core technology required to provide seamless mobility in the wireless mobile networks such as WLAN, 3G Cellular

networks. IP mobility issues can be classified into Location Management and Handover Management. Location management is used to identify the current location of mobile devices and also to keep the track of their location as they move in the network while handover management is to maintain the on-going connections no matter where they move in the network. In this thesis, we will focus on the latter: performance of handover management. The main objective of the handover management is to minimize the service disruption by reducing data loss and handover latency during handover period. [2].

Handover management can be implemented in different layers of the Internet architecture based on the OSI-reference model. The OSI model [3] is based on a proposal developed by the International Standards Organization as the first step toward international standardization of the protocols used in the various layers of communication architecture. The OSI model is designed as a 7-layer hierarchy, with each layer responsible for a specific task as shown in figure1.1.

The two lowest layers handle issues concerning physical medium and raw data transmission. The two highest layers cover presentation and application problems.
Currently, the focus of mobility management is not from these four layers while there are quite a few mobility management protocols developed from the rest three layers.

Network layer provides switching and routing technologies and creating logical paths to transmit data from one endpoint to another endpoint. Internet Protocol (IP) protocol is working this layer to provide routing and forwarding functions , as well as addressing, internetworking, error handling, congestion control and packet sequencing. Most IP mobility researches focused on this layer at the beginning and there are many brand new protocols in this layer which can support IP mobility management such as Host Identity Payload (HIP), Handoff Aware Wireless Access Internet Infrastructure (HAWAII), Cellular IP, and Intra-domain Management Protocol (IDMP). [4] [5] [6]. The most

significant protocol in this layer for mobility management is mobile IP (MIP) and it has been used very commercially and widely. We will describe MIP in Chapter 3.

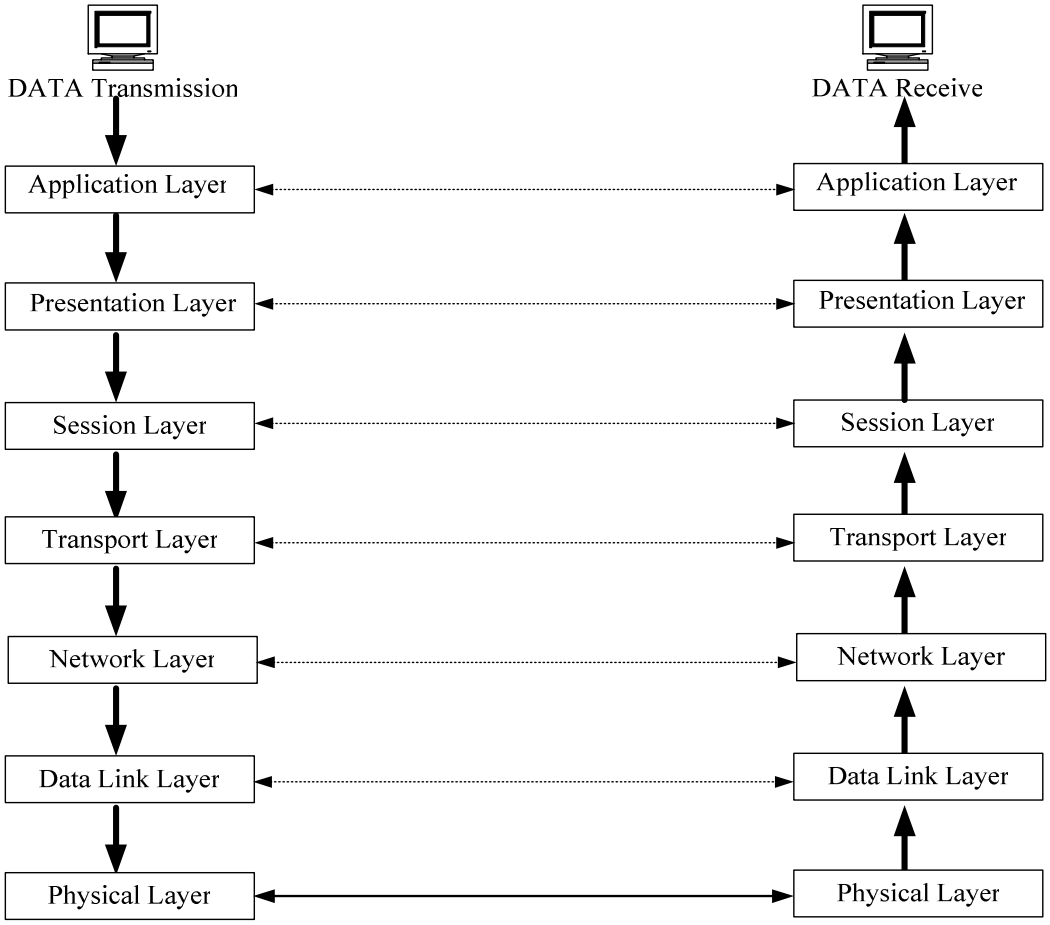


Figure 1.1 OSI 7 Layers Model

The function of transport layer is to provide transparent transfer of data between two endpoints. Moreover, it is responsible for end-to-end error recovery and flow control. Stream Control Transmission Protocol (SCTP) is a new Internet Engineering Task Force (IETF) proposed standard protocol for the transport layer. It is the third protocol for

transport layer and it is designed to eventually replace the other two transport protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Like TCP, SCTP is reliable but offers new features such as multi-streaming and multi-homing. In particular, the Dynamic Address Reconfiguration (DAR) extension of SCTP enables the SCTP protocol to add, delete, and change the IP addresses during an active SCTP association and make SCTP use for handover management. We will present SCTP and Mobile SCTP (mSCTP) in Chapter3.

Session layer establishes, manages and terminates the connections between the applications running on the endpoints. During the session establishment, the endpoints need to exchange the location information. Accordingly, when a mobile device in a session moves to other location, it will send the new location to the other endpoint to establish a new session. Based on this, Session Initiation Protocol (SIP) has been designed recently to support IP mobility. The core idea of SIP will be seen in Chapter 3.

Code Division Multiple Access 2000 (CDMA2000) and 802.11b Wireless Local-area Network (WLAN) are the two wireless technologies used for the project. CDMA2000 network handles mobility by performing step-by-step soft handover mechanism to reselect the base transceiver stations while WLAN uses re-association process to deal with mobility issues. However, there are no standard mechanism defined for handover between CDMA2000 network and WLAN. Using the new features of mSCTP to perform the handover between CDMA2000 and WLAN is the main purpose of this thesis.

The research is organised into 6 chapters.

Chapter 1 provides an introduction to the research as regards the background for the research area as well as the objective of the research.

Chapter 2 gives an overview of CDMA and 802.11 WLAN in terms of the basic

network architecture, mobility management and their differences.

Chapter 3 concludes the advantages of mSCTP when using as the mobility management protocol by comparing mSCTP,MIP and SIP.

Chapter 4 details the seamless handover procedure between CDMA2000 and 802.11 WLAN. A theoretical analysis of the seamless handover is also presented in this chapter in terms of handover delay, end-to-end throughput and packet loss.

Chapter 5 sets up a testbed and designs the software to perform the seamless handover.

Chapter 6 provides the handover results and discusses the findings.

Chapter 2
Wireless Access Interfaces -- CDMA2000 and 802.11b WLAN

IEEE 802.11 WLAN has gained increasing recognition over wired networking in recent years and it makes the communications world progressively more mobile. On the other hand CDMA2000 is a significant advance in the evolution of cellular wireless telecommunications into 3G networks. Both of these two technologies employ wireless access air interfaces for data communication; however they have vastly different characteristics. This chapter describes these two wireless technologies in terms of network architectures and highlight differences between them.

2.1 Overview of 802.11b WLAN

802.11 standards refer to a family of specifications developed by the Institute of Electrical and Electronics Engineers (IEEE) for wireless LAN technology. 802.11 standards specify the physical (PHY) and Medium Access Control (MAC) protocols of the over-the-air interface between a wireless client and a base station or between two wireless clients. 802.11b is a specification of the 802.11 family. 802.11b is also regarded as 802.11 High Rate or WiFi and it is an extension to 802.11 that applies to wireless LANS and provides up to 11 Mbps transmission rate in the 2.4 GHz band.

An 802.11b WLAN network is based on a cellular architecture where the system is subdivided into cells and each cell forms a basic service set (BSS). Each cell is controlled by a Base Station or an Access Point (AP). A distribution system is used to

integrate multiple BSSs to form an extended service set (ESS). A portal device acts as a bridge to connect the WLAN to the wired network such as the Internet. [7] [8]. Fugure2.1 shows the basic network architecture of an 802.11b WLAN.

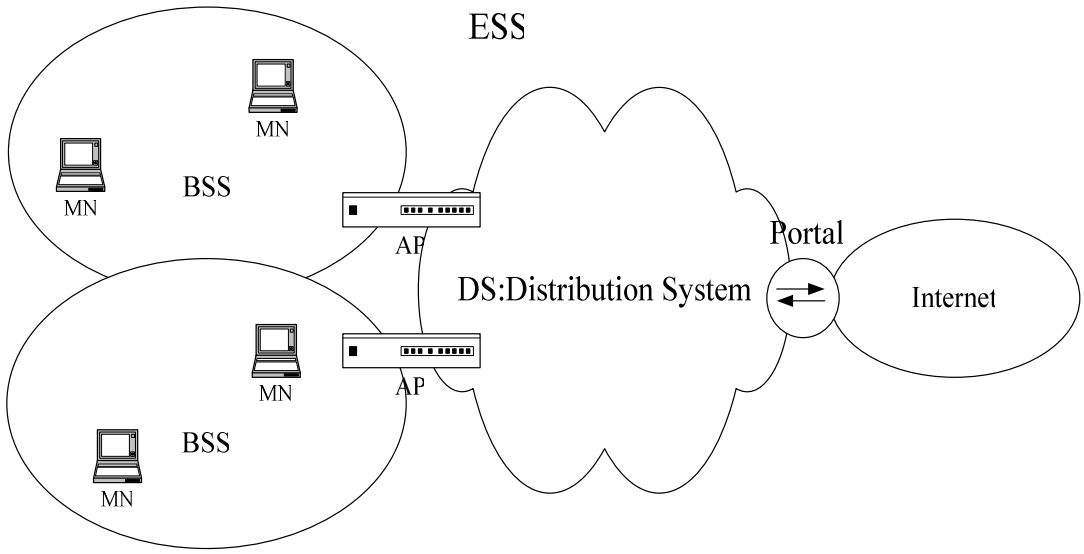


Figure 2.1 802.11b WLAN network architecture

In figure 2.1, the basic components of an 802.11b WLAN are the BSSs and each BSS is formed by several mobile nodes and an AP. The mobile nodes are computing devices equipped with 802.11b wireless network interfaces. The AP is used to control the traffic among the mobile nodes by controlling the wireless network interfaces of each mobile node. There are four control functions of an AP for a BSS:

1. It provides the same 802.11 radio interface protocol for all the mobile nodes. Hence, all the mobile nodes use the same wireless medium channel and they can communicate each other.

2. It also implements the 802.11 Carrier Sense multiple Access with Collision Avoidance (also known as CSMA/CA) MAC protocol to avoid traffic collisions among the mobile nodes.

3. It plays a role as a buffer to store-and-forward to packet to co-ordinate communications among the mobile nodes in a BSS.

4. When mobile nodes communicate some device outside the BSS, the AP has a bridge function to connect the mobile nodes with the device.

After a mobile node in a BSS is on, it starts the scanning procedure to obtain the synchronization information to identify the BSS. The scanning procedure can be either passive or active. If the mobile node only waits to receive a Beacon Frame containing the necessary information from an AP, this is passive scanning. On the other hand, if the mobile node sends a probe request to find the AP, this is active scanning. Passive scanning can save battery power while active scanning is more efficient.

A scan report is generated when the scanning procedure is completed and all the available APs are listed on the report. The mobile node then chooses one of the APs and joins the BSS which the AP belongs to. The procedure is known as authentication process. For the authentication process, the mobile node and AP must exchange the state information and prove the knowledge of a given password. The mobile node can authenticate with several APs at a time.

After the authentication process, the mobile node does not belong to the BSS completely. The final step—association process is required to allow the DS knows the current location of the mobile node. Unlike authentication process, the mobile node can only associate with one access point at a time.
After choosing the AP with the best signal, the mobile node sends an association request to the AP to start the process and the AP responses the request with a positive message containing an association ID. Hence, the AP stores all the information of the mobile node and it starts to process the messages for the mobile node. After the association process, the mobile node can transmit or receive data through the AP.

When the mobile node moves from one BSS to another BSS, the mobile node needs to move the association from the old AP to a new AP by re-association process. During the mobile node's movement, the mobile node monitors the signal from its current AP as well as other available APs in the same ESS. When the signal quality from other APs is better than the current one, the mobile node recognizes that the current AP should be changed and it starts the re-association process by sending a re-association request message to the new AP which has the best signal.

After receiving the re-association request, the new AP first subtracts the address of the old AP from the request message and then it communicates with the old AP to check whether the previous association exists or not. If the old access point does not verify that it authenticated the station, the new AP sends a de-authentication message to the mobile node and ends the re-association process. Thus, the re-association fails and the mobile node can not change the access point.
While if the previous association is authenticated by the old AP, the new AP will send a positive re-authentication message containing a new association ID to the mobile node to inform the change. Simultaneously, the new AP notifies the association change to the old AP. Consequently, the old AP sends all the data in the buffer for the mobile node to the new AP and the new AP starts to process all the traffic for the mobile node. [7][8][9].

By now, the mobile node can use the AP as a bridge for data communications. To sum up, there are four main steps for a mobile node to access a new AP when the mobile node is moving inside an ESS: scanning, authentication and re-association. These process in 802.11WLAN is smoothly and seamless. The whole procedure for a mobile node to access an AP in a BSS and change the can be seen in Figure 2.2:

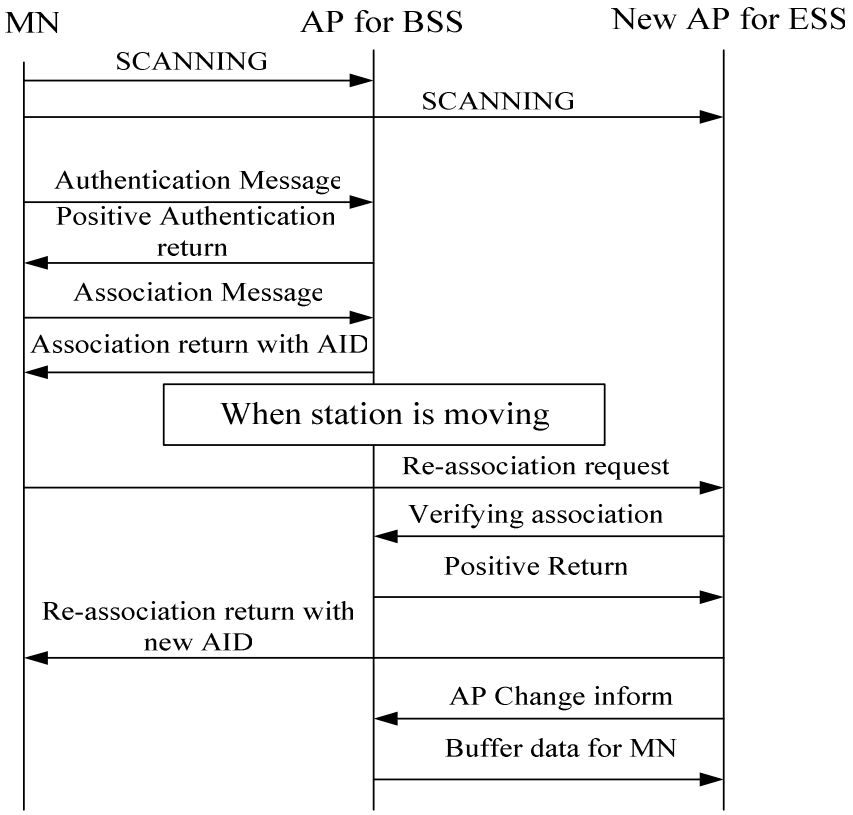


Figure 2.2 Network attachments in 802.11b WLAN

2.2 Overview of CDMA2000

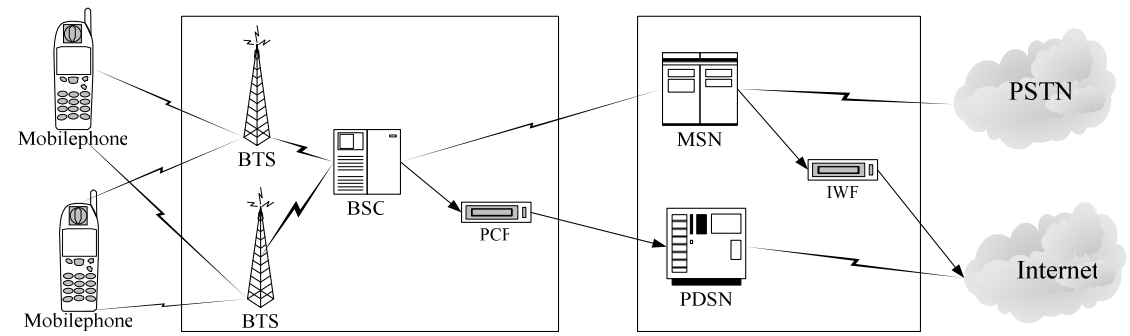
CDMA2000 standards have evolved from CDMA one by Third Generation Partnership Project 2 (3GPP2). They refer to a family of 3G mobile telecommunication technologies using CDMA to send voice and data between mobile phones and base stations. Unlike TDMA (time division multiple access), CDMA is a mobile digital radio technology that permits many radios to share the same frequency channel. By this, CDMA provides added capacity to multiple users without requiring more resource.

CDMA2000 1x, also known as 1xRTT, is the core standard of the CDMA2000 standard family. It is symbolic of the transition of CDMA technology from the 2G to the 3G

telecommunication world. The CDMA2000 1x specification provides three types of data service: short message service (SMS), circuit switched data, and packet switched data. SMS provides two-way pager-like functionality, allowing short text messages to be sent, received, and acknowledged. Circuit-switched data allows dial-up modem connections over the cellular network. Packet-switched data provides Internet Protocol connectivity.

Circuit-switching use the traditional TDM circuit to transmit data and it has the maximum rate of 19.2Kbps while packet-switching provides a point-to-point connection (PPP) between a mobile station and the packet data serving node (PDSN) and can offer data rate up to 144Kbps. [10].In our project, a PPP connection is set up between the mobile node and the server by a CDMA2000 1x Gtran modem provided by Telecom New Zealand.

The CDMA20001x network consists of three major parts: the mobile station, the radio access network and the core network. Figure2.3 is the basic architecture of CDMA20001x networks.



BTS: Base TransceiverStation
BSC: Base Station controller
PCF: Packet Control Function
PDSN: Packet Data Serving Node
MSC: Mobile Switching center
IWF: Interworking Function

Figure 2.3 The architecture of CDMA2000 system

The mobile station refers to a variety of mobile devices such as mobile phones, PDAs and laptops. These devices contain a CDMA2000 radio terminal used for radio communication with radio access network over the radio interface.

The radio access network handles all radio-related functionalities and it provides radio bearers between the mobile station and the core network for the transport of user data and non-access stream signaling. The main components of a radio access network are base transceiver stations, packet control function and base station controller. The base transceiver station is an entity which provides transmission capabilities between the network and the mobile station.

The BSC is an entity which provides control and management for one or more BTSs. It also provides a router function for voice and circuit-switched data between a mobile station and the mobile switching center (MSC). The packet control function routes the packet-switched data between the base station controller and the packet data serving node because the base station controller can not route it to the PDSN directly.

The core network is responsible for switching and routing data messages between the radio access network and external networks such as Internet and PSTN. The main components of the core network are MSC, packet date serving node and interworking function.

The PDSN acts as a gateway between the radio access network and the external network such as the Internet. It provides an IP address to the mobile station and route the IP packets to the external network.

The MSC is responsible as a switch and a database for setting up, terminating, maintaining and forwarding the voice and circuit-switched data. Interworking function plays the role of a bridge between the circuit-switched based voice network and the packet-switched based network. [10][11].

When a mobile station transmits data using CDMA2000 1x, it first needs to register for packet data services and so sends an origination message to its BSC. The BSC replies with a base station acknowledgement to the mobile station after it acknowledges the origination message. Next, the BSC sends a service request message to the MSC to ask for radio resources. After the MSC returns an assignment request message to the base controller, the mobile station starts to set up radio resources.

After the mobile station obtains the radio resources, it still can not transmit the data because the packet-switched data can not be routed to a PDSN. Hence, the PCF in the radio access network selects a PDSN for the mobile station for this data call and sends the PDSN a registration request and waits for the positive return by registration reply sent by the PDSN. Now, the connection between the mobile and the PDSN has been established completely and the data can be transmitted between the mobile station and the PDSN. Finally, the base station controller sends an assignment complete message to the MSC notifying the connection establishment. [10]

The whole procedure to set up a data call in CDMA2000 1x network can be seen in Figure 2.4.

When a mobile station moves between the areas covered by different BTSs in a CDMA2000 1x network, the network provides a soft handover mechanism for mobility management. To do this, CDMA2000 1x network defines 4 sets to store the information of BSs according to the signal strength of the BSs. The first is the active set containing the BSs which the mobile station may simultaneously contact. The second is the candidate set which stores all the BSs not in the active set but which the mobile station can demodulate. The third is the neighbor set for the BSs which are in the vicinity of the BSs the mobile station is using currently. The last is the set of all remaining BSs.

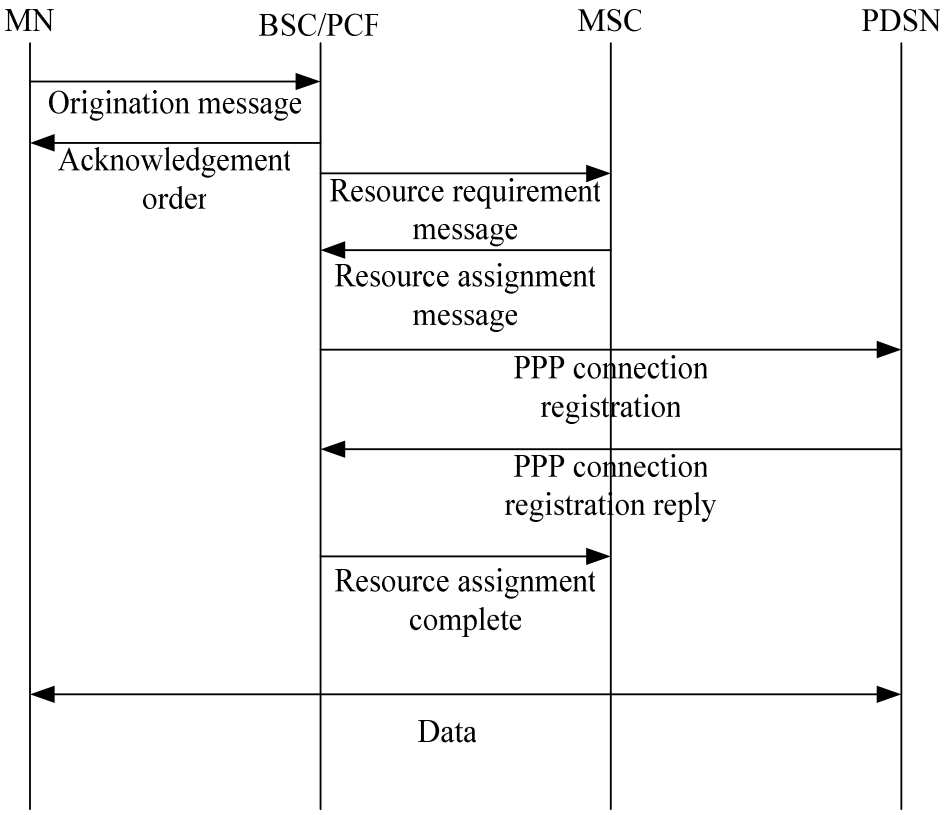


Figure 2.4 CDMA2000 1x network data call setting up procedure

Consider a mobile station using BTS1 and moving toward to BS2. When the mobile station detects that the signal strength of BTS2 exceeds the minimum threshold T_ADD, it initially moves BTS2 to the candidate set because the total received signal strength is not sufficient. The threshold T_ADD is dynamic and defined by a linear function of the signal strength of the total active set. When the strength of BTS2 is detected to be above the threshold, the mobile station informs this event to the network and waits for the handover direction message from the network requesting it to add BTS2 to the active set. Afterwards, the mobile station operates the soft handover.

On the other hand, when the mobile station moves away from the BTS1 and the signal strength of BTS1 drops below the T_ADD, the mobile station starts a handover drop

timer. When the timer expires, the mobile station receives a handover message from the network and move BTS1 from the active set to the candidate set. When the signal strength of BTS1 keeps reducing, the mobile station starts another handover drop timer to move BTS1 from the candidate set to the neighbor set. Hence, the CDMA2000 1x network performs a step-by-step soft handover mechanism with multiple thresholds and timers [11].

2.3 WLAN and CDMA2000

The two previous sections describe some fundamental characteristics of 802.11b WLAN and CDMA2000 in terms of network architecture. This section will be more specific on how these two network technologies resemble and differ by comparing them. Table 2.1 shows the main differences between CDMA2000 and WLAN network communication technologies.

The first remarkable difference between these two network technologies are the range of supported service. All the services supported by 802.11b WLAN are available by CDMA2000.The CDMA2000 standard supports a variety of circuit-switched and packet-switched services including voice, video applications, SMS (short message service), MMS (multimedia messaging service), email, fax, while the 802.11b WLAN only supports the corresponding packet-switched services. Coverage also differs heavily between CDMA2000 and WLAN networks. A CDMA2000 network is based on the cellular concept. Although a base station only covers a small area, a large area can be divided into many small areas or cells each covered by a separate base station.

Characteristics	CDMA2000	WLAN
Services	Circuit-and packet- switched services	Packet-switched services
Data rates	Ranging from 144kbps to 4.8Mbps	Up to 54Mbps
Coverage	National/international coverage	Local coverage
Power control	Flexible power control	Max. effect of 100mW required
Mobility	High	Low

Table 2.1 CDMA2000 and 802.11 WLAN Comparisons

There are two benefits of this concept. Firstly, the mobile station can use the same frequency in one cell as the one other mobile station use in another non-overlapping cell. Secondly, the transmitter power can be lowered by reducing the base station covered area. Also, CDMA2000 1x provides soft handover mechanism so the mobility of the mobile station is high. On the contrary, 802.11 WLAN incorporates a non-cellular concept in terms of much smaller locally situated network islands, the so-called hot spots. Usually these 802.11 WLAN hot spots areas only cover homes, small enterprises, campuses, hotels, hospitals, airports etc, so the coverage of 802.11 WLAN is local. Also, WLAN provides re-association process to make the mobility of the mobile node low. Hence, the advantages of CDMA2000 network comparing to WLAN are more available service, increased capacity, reduced transmitter power and huge coverage.

Another significant difference is in the data rate. CDMA2000 network supports data rates up to144kbps while 802.11b WLAN is much faster than CDMA2000 and can support data rates up to 11Mbps. The costs of deploying these two network technologies

are also very different. A CdMA2000 network is based on cell and each cell needs a BTS, which makes CDMA2000 network so costly. Also, the license and frequency use fees are very expensive. These two aspects make the deployment of CDMA2000 networks hard. Normally, building a CDMA2000 networks needs the permission of a government and is run by a large company. In contrast, the 802.11 WLAN deployments require relatively cheap access points and there is no license or frequency fee for the 802.11 WLAN. It is easy for an individual to create a WLAN at home. Hence, the advantages of WLAN network are low cost and high speed in comparison to CDMA2000 network.

To sum up, both WLAN and CDMA2000 have their strengths and weaknesses. WLAN provides much higher data rates, however the mobility of WLAN is low with only over small areas, whereas CDMA2000 offers relatively lower date rates but much higher mobility with reliable nationwide coverage. As a result, by combining them, it may be possible to achieve high data rates within certain hot-spots, while still obtaining reliable connection outside of the hot-spots. How to ensure that mobile users use WLAN to connect to the internet within hot-spots and switch to CDWA2000 when WLAN is not accessible thus becomes crucial to the evolution of mobile world.

Chapter 3
Mobile SCTP and its Advantages for Mobility Management

In this chapter, we describe three mobility management solutions: a network layer approach, Mobile IP (MIP) including MIPv4 and MIPv6, a transport layer approach, mSCTP and an application layer approach, Session Initiation Protocol (SIP). Firstly, the new features of SCTP and the DAR extension of SCTP are covered to examine mSCTP handover mechanism. Then, the basic mechanism and handover procedure of MIP and SIP are described. At the end, a comparison of these three mobility protocols shows the advantages of mSCTP for mobility management.

3.1 Mobile Stream Control Transmission Protocol (mSCTP)

Mobile SCTP is the transport layer handover solution based on the multi-homing feature of SCTP and its dynamic address reconfiguration extension (DAR). Multi-homing is the concept that SCTP endpoints can hold multiple IP addresses while DAR extension provides mSCTP a method to reconfigure the status of IP address on an ongoing SCTP association.

3.1.1 Stream Control Transmission Protocol (SCTP)

SCTP is another general-purpose transport protocol for IP network data communications and it is regarded as the third transport protocol after TCP and UDP.

TCP and UDP have been deployed for the last 20 years and they have been as the major transport protocols for most of the networks. However, both of TCP and UDP have shortcomings. Some commercial applications such as real-time multimedia and telephony applications already require greater functionality than what TCP and UDP can offer. To extend transport layer functionality, the IETF published SCTP in October 2000 as a proposed standard. [12][35].

Although TCP provides reliable data transmission service between two communication endpoints like SCTP, the following limitations make it difficult to meet requirements of new commercial applications [13][14]:

1. All the data should be transmitted in strict order between two endpoints when using TCP as the transport protocol. This is too restrictive for applications with partial ordering or no sequential delivery. Moreover, loss of a single TCP segment can block delivery of all subsequent data in a TCP stream due to the strict sequence maintenance. All the subsequent data can only be transmitted after retransmit the lost TCP segment and this introduces some unnecessary delay to the overall data. This is termed head-of-line blocking.
2. The byte-oriented nature of TCP treats each data transmission as an unstructured sequence of bytes, so applications must add their own record marking to delineate their messages. Also, applications must use the push mechanism to ensure a complete message is transferred in a reasonable time.
3. TCP does not support multi-homed IP endpoints. An application can only bind a single IP address to a particular TCP connection with another endpoint. There is no backup or secondary IP link that can be used when a link failure occurs in

TCP. Hence, link or path-level redundancy is a big challenge and the data transmission service becomes vulnerable to link failures.

4. Some security problems can be introduced due to the three-way TCP connection setup handshake. TCP is known to be susceptible to some denial of service attacks, such as the "SYN flooding" attack.

Not just TCP, UDP has its own shortcomings as follows when it is considered to be transport protocol for some particular applications such as telephony signalling. [13]

1. UDP is a message-oriented protocol and it only provides an unreliable data transmission service to the application. Accordingly, the sender does not know whether the receiver receives the data or not. Moreover, there is no guarantee on the ordering of the data sent to the receiver in UDP, so the sender may send one single UDP segment more than once and the receiver may receive more than one copy of the UDP segment.
2. There is no built-in congestion management mechanism to detect path congestion in UDP. This may cause either more data is injected into the network or more data is discarded by the network when the network is congested. Hence, UDP becomes more unreliable when the network is congested.

SCTP is also message-oriented but provides a reliable data transmission service. As a new transport layer protocol, it combines the advantages of TCP and UDP. More importantly, SCTP defines some new features as follows to overcome the drawbacks of TCP and UCP in order to support some new commercial applications. [14][15]

1. **Unordered data transmission**. Data can be transmitted by multiple modes including strict order like TCP and unordered delivery like UDP. Partially ordered delivery is also possible in SCTP. Different applications can choose different orders to transmit the data according to their requirements.
2. **Multi-homing support.** SCTP endpoints can have more one IP address so that the sender can reroute packets to the receiver using a backup IP address when the

current IP address becomes unreachable. This makes SCTP tolerant to link or path failure.

3. **Multi-stream support.** SCTP uses multi-stream to prevent "head-of-line blocking". When loss of a single SCTP chunk occurs on one stream, the retransmission only affects that stream while it does not affect other streams.
4. **Security cookie against SYN flood attack.** SCTP employs a security mechanism- cookie to mitigate the impact of TCP SYN flooding attacks. Each time when the endpoints initiate a SCTP association, they exchange and confirm the cookie information.
5. **Built-in heartbeat mechanism.** SCTP employs heartbeat mechanism to obtain the information of every possible IP address. SCTP endpoints send a periodical message to all the potential IP addresses to check the status of IP addresses and SCTP endpoints treat an IP address inactive once it reaches the threshold of unreturned heartbeat acknowledgements.
6. **Message boundary preservation.** SCTP design chunk to be its data structures. A chunk can preserve applications' data frame boundaries no matter what the size of the data frame. Thus, multiple messages can be bundled into a single chunk while a large message can be handled by multiple chunks. This makes SCTP more flexible to store the data frame of the application.

When SCTP was originally designed, it was only considered to be a transport protocol to support some new applications. Mobility wais not considered to be an issue for SCTP. However, the multi-homing feature enables a SCTP mobile node having a backup IP address to replace the primary IP address. This makes it possible to handle mobility at the transport layer. The following section describes multi-homing in more detail.

3.1.2 SCTP endpoints and association: multi-homing

A SCTP multi-homing endpoint uses multiple IP addresses to make an association to the peer SCTP endpoint. Of these IP addresses, only one IP address is considered as the primary IP address while all the other IP address are used as the backup IP address. Once the endpoint detects some errors on the primary IP address, it will choose one of the backup IP address to be the primary IP addresses. The new primary IP address still uses the same SCTP port as the old primary IP address. Therefore, a SCTP multi-homing endpoint can be presented as a list of IP addresses on the device that share a single SCTP port. [13]

Figure3.1 shows some examples of multi-homing endpoint.

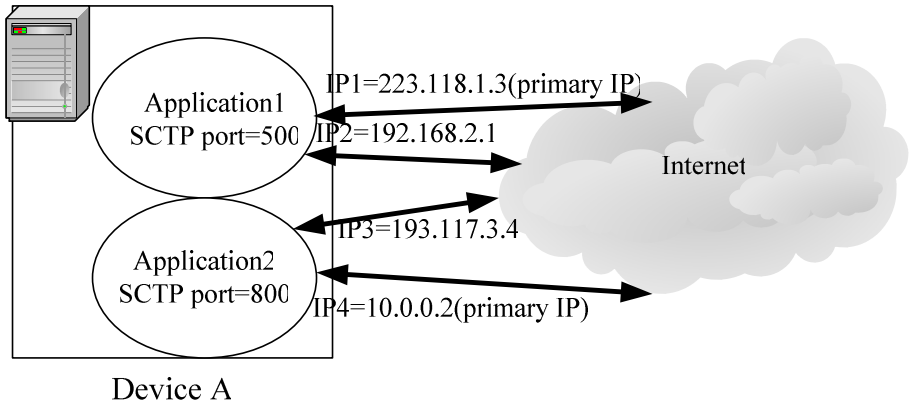


Figure 3.1 Multi-homing endpoints

In Figure 3.1, there are 2 SCTP multi-homing endpoints for device A. The first is endpoint1 using IP address1 and IP address 2 with SCTP port 500, that is:

endpoint1= [223.118.1.3, 192.168.2.1:500]

The other is endpoint2 using IP address3 and IP address4 with SCTP port 800, that is

endpoint2= [193.117.3.4, 10.0.0.2:800]

The primary IP address of endpoint is IP address 1. When IP address 1 is not reachable, only IP address 2 can replace it to be primary IP address for endpoint1. The same principle works for endpoint 2.

For two multi-homing endpoints, there is only one SCTP association between the endpoints. The association uses the primary IP addresses of the multi-homing endpoints as the sender and destination IP addresses. Figure3.2 shows an established association between two multi-homing SCTP endpoints:

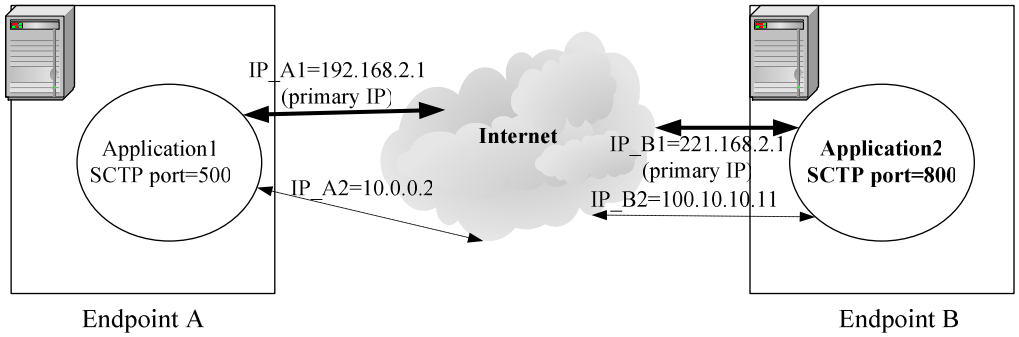


Figure 3.2 Association between two multi-homing endpoints

The association depicted in Figure 3.2 can be described as follows:

Association= [192.168.2.1, 10.0.0.2:500]: [221.168.2.1, 100.10.10.11:800]}

There are 4 possible paths for the association, either through the primary IP address or the backup IP address. Only one path between IP_A1 and IP_B1 is considered the primary path. Endpoint A and endpoint B should always use this path for data transmission unless some specified by the SCTP user. [22]

When a failure occurs on the primary IP address of a multi-homing endpoint, the endpoint knows the primary IP address is not reachable and choose the backup IP address to be the primary IP address. Hence, it is very crucial for the SCTP endpoint to keep and update the status information of each IP address. The heartbeat mechanism is designed in SCTP to help the implementation of multi-homing of SCTP. In the heartbeat mechanism, heartbeat chunks are sent periodically to the destinations through the idle paths (all the paths except the primary path), and a counter is maintained of the number of heartbeats chunks sent to an inactive destination without receipt of a corresponding acknowledgment. When the number exceeds the set up threshold, the destination IP

address will be treated inactive. For instance, in Figure 3.2, endpoint A uses IP_A1 as the primary IP address and the endpoint B keeps sending heartbeat chunks to the endpoint A to check the state of IP-A1 and IP-A2. At some point, if the count of no replies from IP_A1 exceeds the set up threshold, then the endpoint B will treat IP_A1 inactive and the endpoint A becomes single-homing. The primary IP address of the endpoint switches to IP_A2.

3.1.3 Dynamic Address Reconfiguration (DAR) Extension

Using the multi-homing feature, a SCTP endpoint can use multiple IP addresses for an association with other endpoint at a given moment. The problem is that basic SCTP does not have a function to add a newly coming IP address to the existing association or change the backup IP address to be primary IP address without disconnecting the endpoint. These problems motivated the DAR extension of SCTP. DAR extension defines three major parameters to change the status of IP address for an association: Add IP Address (Add-IP), Set Primary IP Address (Set-Primary-IP) and Delete IP address (Delete-IP). Also, two special chunks are defined in the DAR extension to carry these parameters between SCTP endpoints. Thus the DAR extension provides the following function to an SCTP association [16]:

1. Dynamic addition of a newly coming IP addresses to an existing association.
2. Dynamic deletion of IP an old and unused addresses from an existing association.
3. Change the primary IP address of an existing association.

3.1.3.1 DAR extension parameters

The DAR extension functions are provided by three new parameters. A SCTP endpoint can add a new IP address to the association set up already by sending an Add-IP parameter to the peer endpoint of the association. Delete-IP parameter is used to delete a backup IP address from the association when a endpoint determines the IP address is not

accessible by heartbeat mechanism. Set-Primary-IP parameter is sent by a SCTP endpoint when it needs to replace the current primary IP address by a backup IP address. The message formats of these three parameters are shown in Figures 3.3, 3.4 and 3.5, respectively

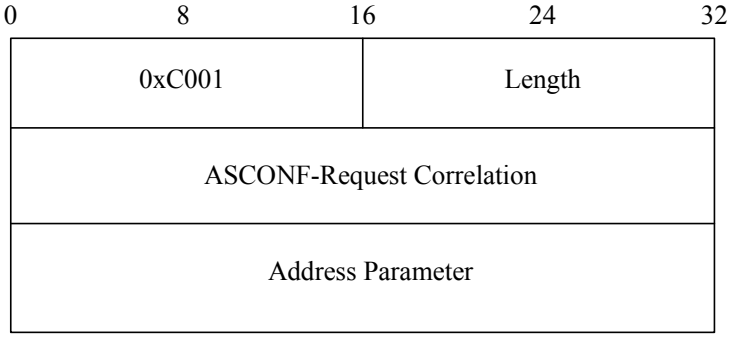


Figure 3.3 Add-IP format

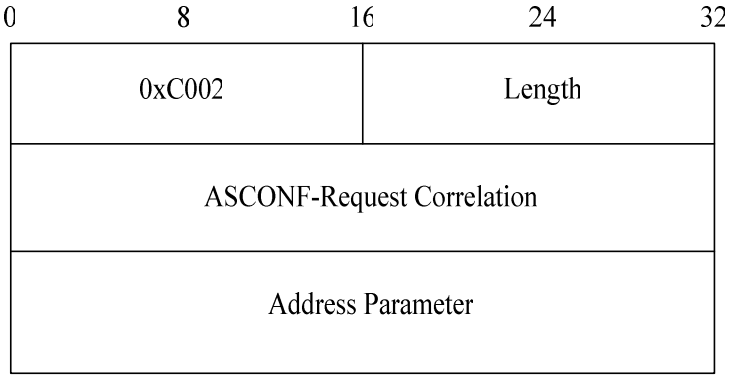


Figure 3.4 Delete-IP format

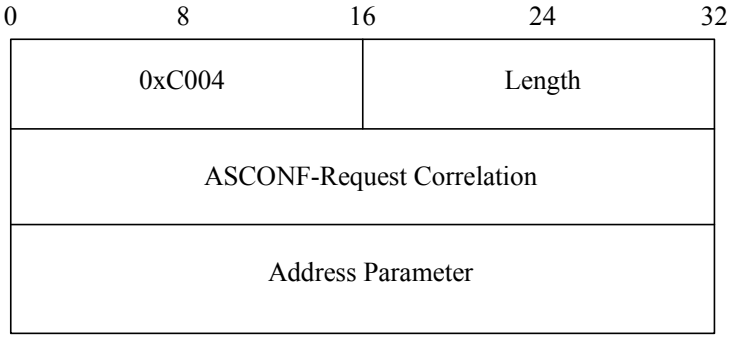


Figure 3.5 Set-primary-IP format

The formats of DAR parameters are almost the same. Four parameters determine the character of the DAR formats.

The first one identifies the type of the parameters. In Figure 3.3, the type value is 0xC001 which identifies that this parameter is Add-IP parameter, while in Figure 3.4, value 0xC002 refers to delete-IP parameter and 0xC004 in Figure 3.5 stands for set-primary-IP parameter.

The second parameter in these formats is the length for the DAR parameters. The value of the length depends on the number of IP addresses carried by the parameter.

The third of these parameters is ASCONF-request correlation ID. This value is the symbol to make the chunks carrying these parameters unique. Hence, the endpoints can distinguish a particular chunk from other chunks. The last parameter in these formats is the address parameter. Address parameter field holds IP address in TLV format. The IP address can be either IPv4 address or IPv6 address in TLV. [12][16].

3.1.3.2 DAR extension chunks

In SCTP, all the data and control parameters are carried by chunk. Chunks are chosen as a basis for SCTP to establish a "building block". Each chunk is composed of chunk type; chunk flags chunk length and chunk data fields. Figure 3.6 shows the structure of a chunk:

Chunk type	Chunk flags	Chunk length
Chunk data		

Figure 3.6 The structure of a chunk

The chunk type field is an 8-bit value and is used to identify the type of the chunk while chunk flags define any special flags used for the chunk and it is also 8-bit. The chunk length is 16bits and it contains an unsigned integer to indicate how long the chunk is. Most of the chunks are specified when SCTP is originally designed but none of them can

be used to deliver DAR parameter. Hence, DAR extension in internet draft defines ASCONF and ASCONF-ACK chunks, two special chunks to carry DAR extension parameters including add-IP, delete-IP and set-primary-IP. Figure 3.7 and Figure 3.8 show the message format of ASCONF chunk and ASCONF-ACK respectively:

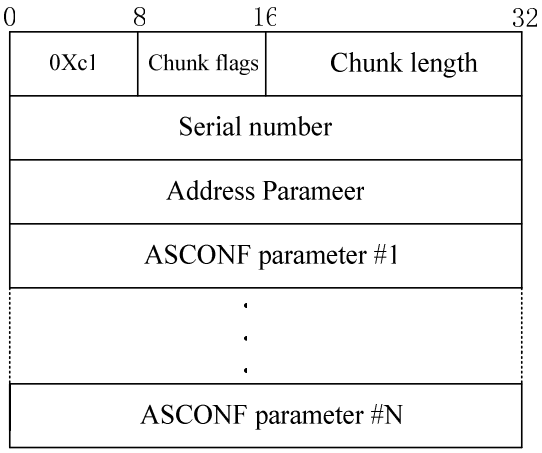


Figure 3.7 Message format of ASCONF chunk

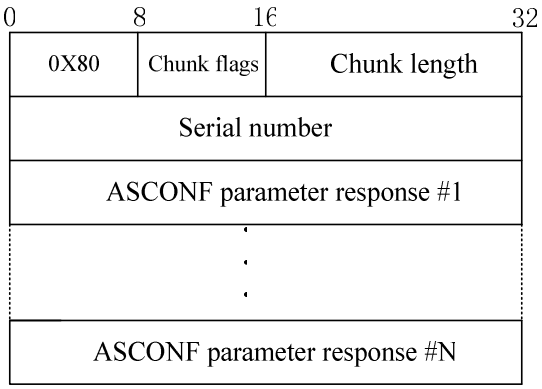


Figure 3.8 Message format of ASCONF-ACK chunk

The formats of ASCONF and ASCONF-ACK chunks are similar to other SCTP chunks. The first value is used to indicate the type of the chunk, 0Xc1 refers it is an ASCONF chunk in Figure 3.7 while 0X80 stands it is an ASCONF-ACK chunk in Figure 3.8.

Chunk flags are set to 0 since chunk flags are not used for ASCONF and ASCONF-ACK chunks. For ASCONF chunk, the serial number is a number randomly generated from 0 to 2*32-1.

For ASCONF-ACK chunk, the serial number must be the same as the serial number produced in its ASCONF chunk address parameter. Then the receiver knows the source of the ASCONF chunk and the primary IP address will be the destination IP address for the ASCONF-ACK chunk automatically. Thus, there is no address parameter for ASCONF-ACK chunk.

An ASCONF chunk is sent by a SCTP endpoint to inform the destination endpoint to make some change to the association. The ASCONF-ACK chunk is sent by the destination endpoint to show the result of the change. Other ASCONF parameters in figure 3.7 refer to the type of function for the parameters and they must be one of the DAR parameters to change the status of the association. The relevant response will appear in ASCONF-ACK chunk in figure 3.8 and it indicates to the result of the ASCONF requests. If the requests succeed, a blank ASCONF parameter response will be returned, otherwise some error indications will be returned. Since DAR extension can change the status of an IP address within an active SCTP association, handover management can be handled by mSCTP now. [12][16].

3.1.4 mSCTP mobility management in WLANs

In this section, we describe the detailed handover procedure using mSCTP when a mobile node moves between two WLANs. One issue of mSCTP mobility management is that it does not support location management. Hence, mSCTP handover is only for the association triggered by the mobile nodes because the MN knows the movement of itself and the signal strength from the old and new access points. The topology for the handover scenario is shown in Figure 3.9. Before the handover, we assume that the MN

is single-homed and has already initiated an SCTP association with the CN using the IP address from area 1. We also assume that the MN always keeps moving forward.

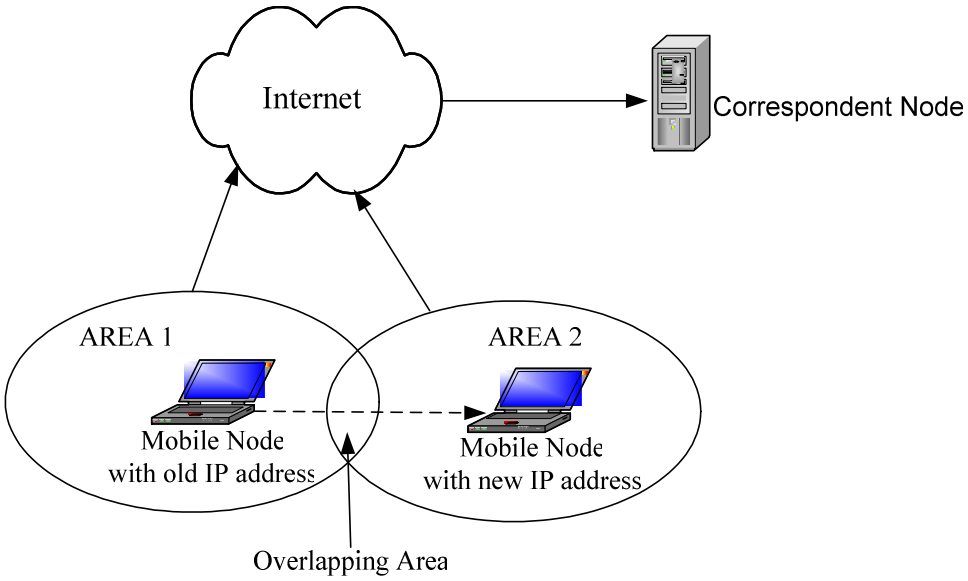


Figure 3.9 mSCTP handover between two WLANs

Then the handover procedural steps may be described as follows:

Step 1) **Obtaining an IP address for the new location:**

When the MN moves to the overlapping area, the MN detects the signal from the AP in area 2 and it obtains a new IP address in area 2.

Step 2) **Adding the new IP address to the SCTP association:**

After obtaining a new IP address, the MN becomes multi-homed and it needs to add the new IP address to the association as the back up IP address. The MN sends an SCTP ASCONF chunk with Add-IP parameter to the CN to initial the change. The CN replies by sending an ASCONF-ACK Chunk to confirm the change.

Step 3) **Changing the primary IP address:**

When the MN keeps moving close to area 2 and it detects that the quality of signal of area 2 is better than area 1, the MN needs to use the better one as the primary path. The MN sends a SCTP ASCONF chunk with Set-Primary-IP parameter to the CN to inform the change and the CN replies with an ASCONF-ACK to confirm the change.

Step 4) **Deleting the old IP address from the SCTP association:**

After the primary IP address has been changed to IP address from area 2, SCTP uses heartbeat chunk to check the status of IP address from area 1. When the MN keeps moving away from area 1, the quality of signal from area 1 is getting lower and its IP address reply to heartbeat chunk will time out. At this point, SCTP association treats IP address from area 1 inactive and MN sends a SCTP ASCONF chunk with Delete-IP parameter to the CN to delete the IP address from area 1 from the SCTP association. The CN replies with an ASCONF-ACK to confirm the change. [2][18].

By now, the handover from area 1 to area 2 is completed. Whenever the MN moves to a new location, this step-by-step procedure for handover management will be repeated. The whole procedure can be seen in Figure 3.10

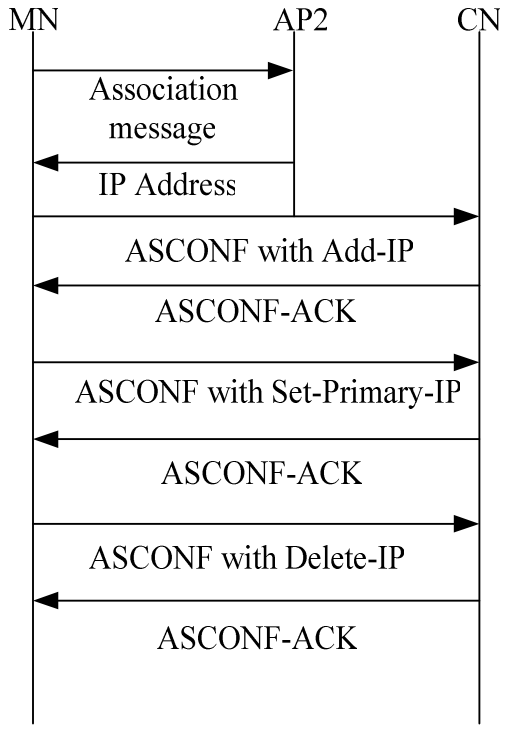


Figure 3.10 mSCTP handover between two WLANs

3.2 Mobile IP

Currently, Mobile IP is the most widely known mobility management protocol and is the most common solution for offering seamless roaming to mobile devices for the internet. Mobile IP is a supplementary network layer mobility protocol defined by IETF, which allows users to stay connected and maintain ongoing applications while roaming between IP networks. [17].

Figure3.11 illustrates the architecture of a MIP network in terms of mobility management.

.

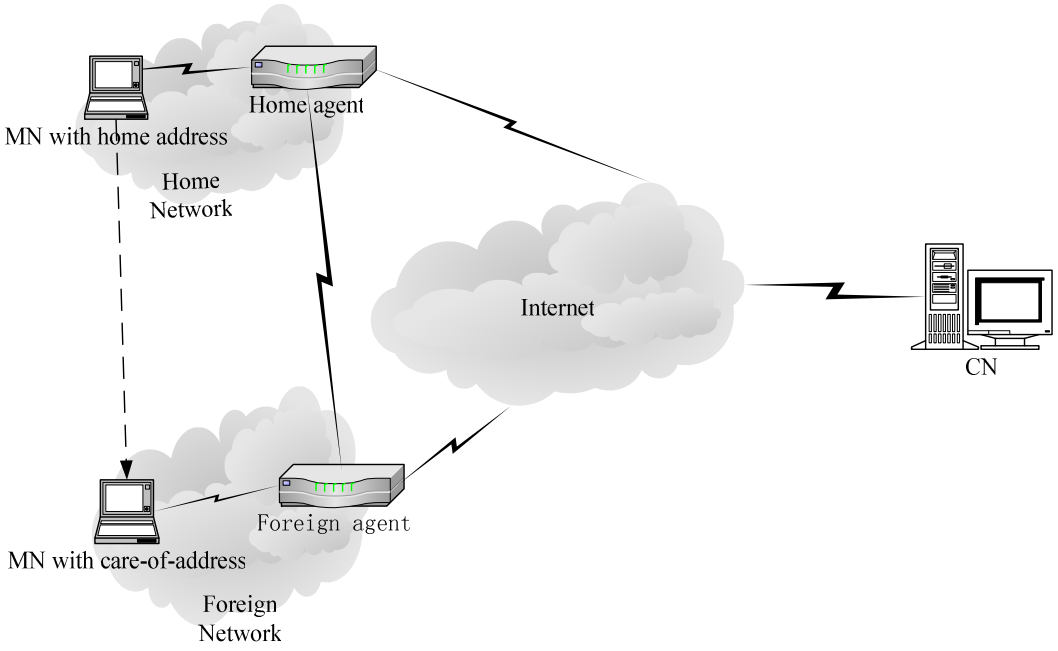


Figure 3.11 Architecture of Mobile IP

MIP defines a few physical components to perform the handover as follow:

Home Agent (HA): It is a router on the home network serving as the anchor point for communication with the mobile node.

Foreign Agent (FA): It is a router that functions as the point of attachment for the mobile

node when it moves to a foreign network and it delivers the packets from Home Agent to the mobile node. It only functions in MIPv4 and it does not exist in MIPv6.

Care-of-address: It is the IP address of the mobile node when it stays at a foreign network

For IP based networks, routing is based on a fixed IP address. However, a problem occurs when a device moves away from its home network and is no longer reachable using normal IP routing. Mobile IP was originally created to allow the mobile node to effectively utilize two IP addresses, one is the home address when the MN is in its home network, and the other is a care-of-address when the MN is in a foreign network. When using a care-of-address for MIP handover, there are there main phases:

1. Discovering the care-of-address in a foreign network.
2. Registering the care-of-address with the HA
3. Tunneling the packets form the HA to the current care-of-address.

When the mobile node is in the home network, it uses its home address to send or receive packets. When the mobile node moves away from its home network and enters a foreign network, it obtains a care-of-address the foreign agent by exchange of agent solicitation and advertisement massages, this is regarded as discovering the care-of-address.

After the mobile node finds its care-of-address from the foreign agent, it sends a registration request to its home agent from its new location to inform the home agent of the mobile node's new care-of-address. When the home agent receives this request, it adds necessary information to its routing table and sends a registration reply message back to the mobile node to approve the request. This is regarded as registering the care-of-address.

Next, the home agent starts to intercept packets destined to the mobile node's home address and tunnels them to the mobile node's care-of-address. This encapsulation changes the destination IP address of the packets from its home address to its care-of-address. When the tunneled packets are received by the foreign agent, they are decapsulated, which changes the destination address back to the mobile node's home address. This is regarded as tunneling the care-of-address. The whole procedure is shown in Figure3.12

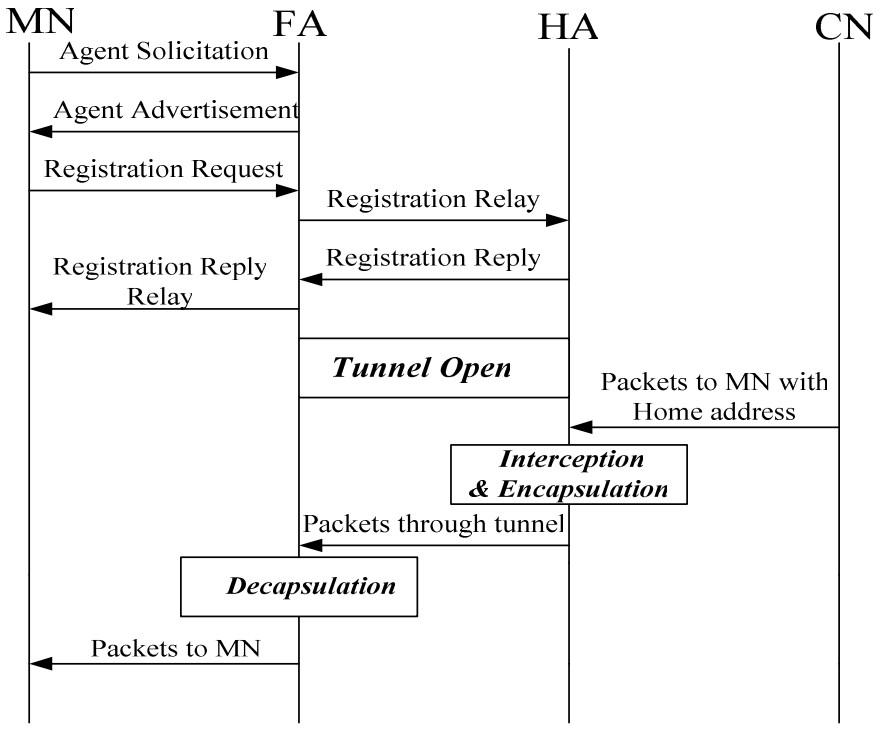


Figure 3.12 The basic operation of Mobile IPv4

When the MN sends packets back to CN from a foreign network, it uses a mechanism called reverse tunnelling. The MN first sends packets back to the HA with the care-of-address as the source address, and then the HA route the packets to the CN with the home address as the source address. However, both tunnelling and reverse tunnelling are far from optimization routing path. Moreover, the IPv4 protocol does not have sufficient

address spaces. To solve these issues, mobile IPv6 is proposed as the natural extension of MIPv4. [32]

In MIPv6, the concept of foreign agent is eliminated. According, the MN can not obtain the care-of-address by listening to the foreign agents advertising like it does in MIPv4. Instead, the MN listens to the router advertisements, which include some extra bits. These extra bits contain the prefix information option format. Consequently, the MN can determine whether it is in its home network or a foreign network from the network prefix in the router advertisement. If the MN finds that the network prefix does not match the home network prefix, the MN determines it is on a foreign network and it starts to obtain a care-of-address. The care-of-address can be obtained through a DHCP server or IPv6 Stateless Address Auto configuration. When the MN obtains the care-of-address, it sends a binding update message to the home agent to inform the home agent its care-of-address. The home agent replies with a binding acknowledgment message. [19][34].

Another alternation of MIPv6 from MIPv4 is that MIPv6 uses route optimization to reduce the traffic and delay of the network. In MIPv6, the packets from the CN are being tunnelled to the care-of-address of the MN by the HA only for the first time. Then, the MN sends a binding update message to the CN to inform the CN its care-of-address. The CN replies with a binding acknowledgment message. Thus, the MN and the CN can communicate with each other without the home agent. [31][36].

We can see the difference between MIPv6 and MIPv4 is that MIPv6 only uses home agent, which can reduce the time to transmit the packets among the foreign agents in MIPv4 to achieve route optimization. The whole process of MIPv6 can be seen in Figure 3.13:

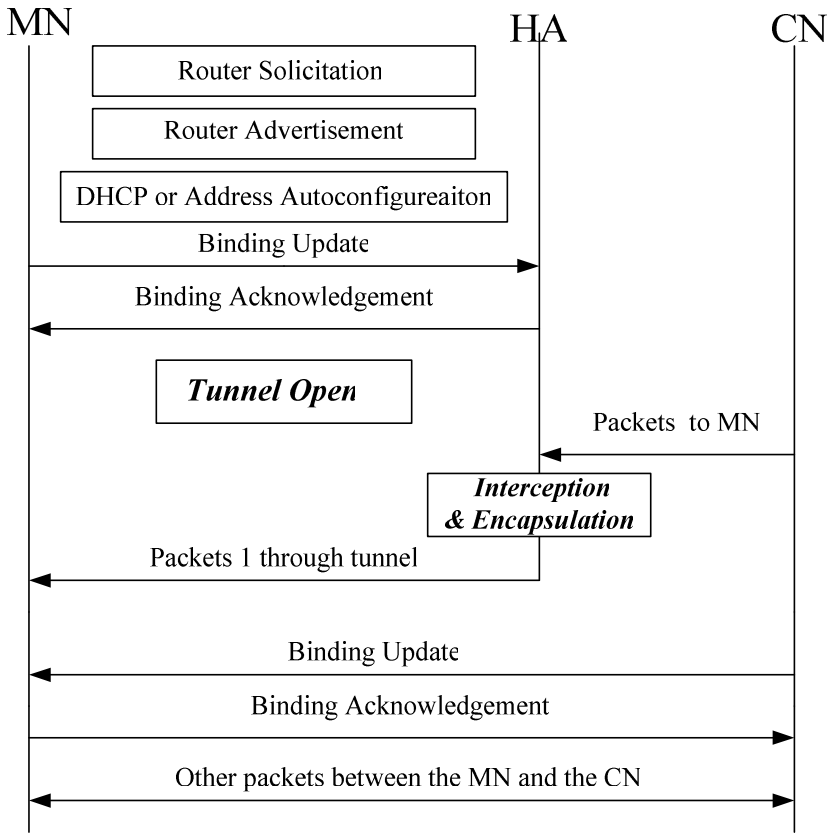


Figure 3.13 The basic operation of MIPv6

3.3 Session Initiation Protocol (SIP)

MIP is known to suffer from high handover delay and it is therefore only suitable for long-lived TCP connection. Real-time applications are sensitive to the network delay and so MIP can not be used for them. Alternatively, SIP was proposed as a handover solution for real-time applications in IP-based network. [20].

SIP is a protocol from the session layer and was originally proposed by the IETF as a general multimedia session initiation protocol establish, modify, and terminate multimedia sessions. In SIP, the endpoints are identified by SIP URIs which is like

email address such as user@host, where user is the name of the endpoint and host stands for the domain name or numerical address.

SIP defines four logical entities: user agent, registrar, redirect server and proxy server. Figure 3.14 illustrates the architecture of a SIP network from a mobility point of view.

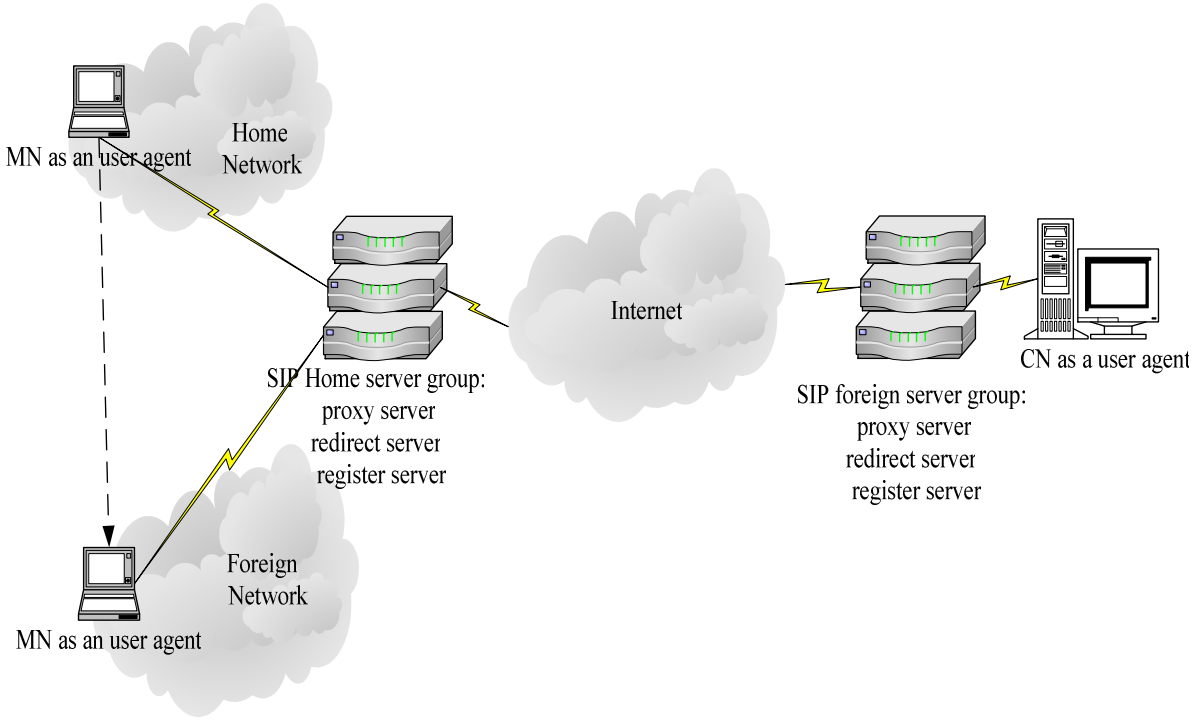


Figure 3.14 Architecture of SIP

A user agent functions either as a user agent client or a user agent server. A user agent client is a client application that issues SIP requests and receives the responses. A user agent server is a server application that receives requests and issues the responses.

Typically, a SIP endpoint functions as a user agent client or a user agent server depending on whether the endpoint initiates the request or not. A proxy server is an intermediate device relaying messages. It receives SIP requests from a client and them forwards them to the next SIP server in the network. A redirect server keeps the track of the MN's location and manages. It also provides the information about next hop or hops

that a message should take and then the MN contacts the next hop device or the CN directly. A registrar server is responsible for processing requests from the MN for registration of its current location.

When the mobile node resides at its home network, it first sends the location information message to the registrar server and the registrar server processes the message and replies with an acknowledgement message to confirm the location information. Now the mobile node can start to establish a session with the correspondent node (CN) through the redirect server. Next, the mobile node sends an invite request to its redirect server in the home server group. The home redirect server forwards the request to the redirect server in the CN's domain. The redirect server in the SIP foreign server group consults the location service to find the address of the CN and sends the address back to the mobile node. After the mobile node sends an acknowledgement of the address message, the session is established. Figure 3.15 illustrates the process of establishing a SIP session.

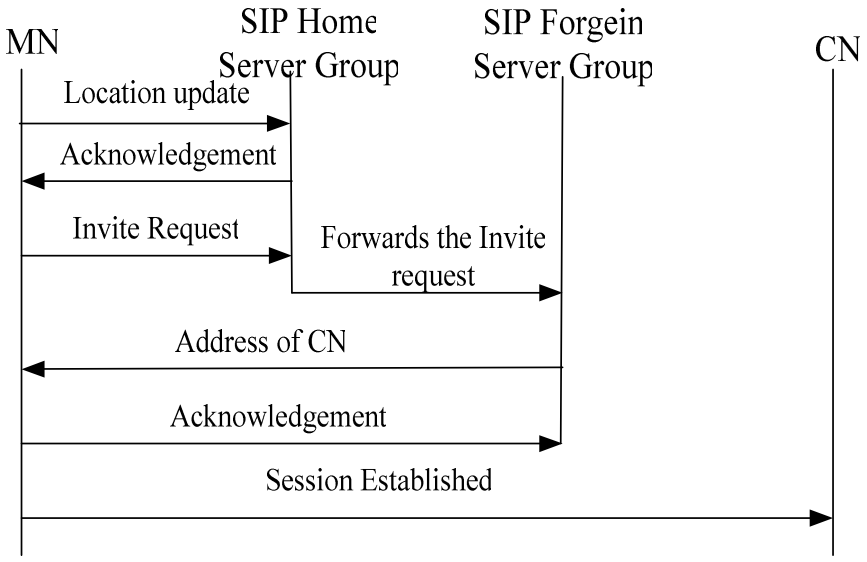


Figure 3.15 A SIP session establishment process

When the MN moves to a foreign network during a SIP session, it first updates its

location by sending a location update message to the registrar server. The registrar stores the information and replies by sending an acknowledgment back to the MN.

After updating the location information, the MN has to send a new invite Request to the correspond node to maintain the session. The new invite message has the same call identifier as in the original session establishment. The request contains the new address of the MN. The CN sends an acknowledgment message to the MN to confirm the address change. Thus, the session can be maintained unaffected. [33][21].

The whole process of how SIP deals with mobility can be seen in Figure 3.16

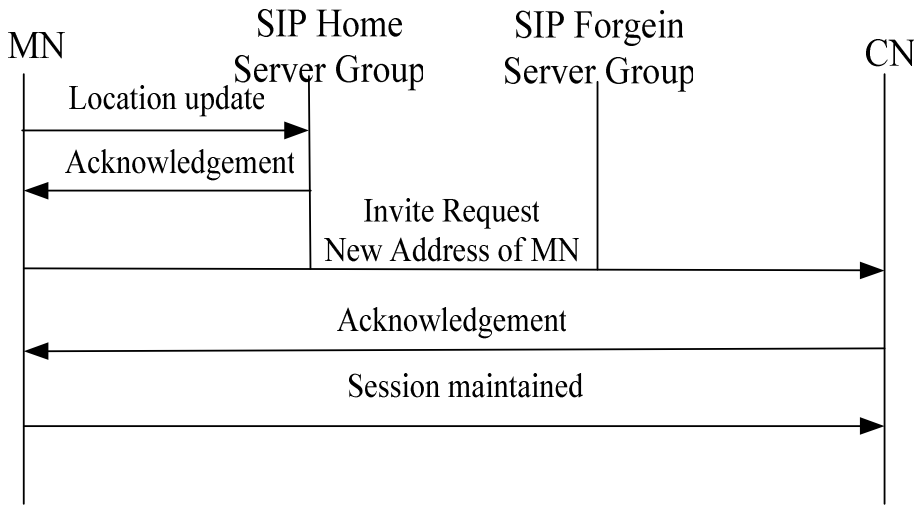


Figure 3.16 SIP mobility management process

3.4 Mobility protocol comparison

The previous sections describe the basic idea of three mobility management protocols and this section compares these protocols to demonstrate the handover advantages of mSCTP.

Table 3.1 summarizes the comparison of mSCTP with MIP and SIP mobility management protocols.

Protocol	MIPv4	MIPv6	mSCTP	SIP
Protocol layer	Network	Network	Transport	Session
Transport Protocol	TCP/UDP	TCP/UDP	SCTP	UDP
Location management	Yes	Yes	No	Yes
Route Optimization	No	Yes	Yes	No
Deployment	Two agents: HA in home network FA in foreign network	One agent: HA in home network	No addition device	SIP server in home network

Table 3.1 Comparison of MIP, mSCTP and SIP

Firstly, we notice that these protocols operate in different layers. MIP handover operates at the network layer. The function of the network layer is to route the packets to the destination through a number of intermediate routers, where all the intermediate routers only forward the packets to the next hop instead of keeping the state of the whole connection. In order to perform MIP handover, not only the MIP endpoints but also the intermediate routers are required to implement MIP protocols.

mSCTP handover is performed at the transport layer. The transport layer is the first end-to-end layer in the OSI Seven Layer Model. It provides a logical path for the application process running on the endpoints. Thus, mSCTP handover ignores the intermediate nodes and only needs that both the endpoints implement mSCTP.

SIP is a session layer protocol. Session layer is also end-to-end and it is responsible for exchanging messages between two endpoints. As a result, it is implemented like mSCTP only at the endpoints.

As can be seen, if the mobility management protocol operates at a lower layer of the OSI model, more network entities are involved in implementing the protocol.

Secondly, the nature of each protocol makes the handover performance differ from each other. A MN needs to complete three steps when performing MIP handover before it can receive packets form the previous point of attachment, which causes high handover latency. In addition, some or all of the packets destined to the MN's old CoA will be lost because the old CoA can not communicate with the MN during the handover period. This may introduce a high packet loss rate. Tunnelling mechanism is used to handle mobility for MIP. This inefficient routing wastes network resources and increases the required processing power of the mobile agents. [23][5].

SIP handles mobility by using the new address to re-invite the CN with the same session identifier as the old session. The session can be continued and it does not produce high delay. However, it is based on an established session and the session is UDP based. That is, SIP only supports UDP applications and this limits the use of SIP. [24]

For mSCTP handover, the endpoint utilizes two network interface adapters to associate with the peer endpoint. When one adapter is down, another adapter is still in the association to make a new path with the peer endpoint. This "make-before-break" handover can reduce handover latency heavily, thereby increasing the quality of the network.

In additional, mSCTP is based on SCTP application. As mentioned before, SCTP is more powerful than UDP and TCP and, indeed, is designed to replace TCP and UDP. From this stand point, mSCTP has a huge potential marketing advantage.

The third difference between these mobility management protocols is in the implementation requirement. The implementation of MIPv4 is the most expensive

because it needs at least two additional devices. A home agent is necessary to work in the home network and each foreign network should have a foreign agent to store the information of the MN when MN moves into the foreign network. The MN, home agent, foreign agents and the CN should all support MIPv4 as well. MIPv6 is similar to MIPv4 apart from that only a home in the home network is needed. For SIP, an additional device SIP server is needed in the home network.

Only mSCTP does not need any additional devices; the only requirement of mSCTP is that both the endpoints should support mSCTP, which make mSCTP less costly and much easier to implement.

To sum up, mSCTP have the following advantages as a mobility management protocol over other protocols:

1. No additional devices required.
2. Easy to implement.
3. Low handover latency.
4. Low packet loss rate.
5. No use limitation.

Due to these advantages, mSCTP is chosen to be mobility management protocol in this project. Because mSCTP does not support location management, only a connection initialled by the MN is considered in the project.

Chapter 4
Mobile SCTP Mobility Management between CDMA2000 and 802.11 WLAN

In chapter 1, we described the different characteristics of WLAN and CDMA2000 networks. In chapter 2, we considered mSCTP handover solutions and their relative advantages. In this chapter, we present the mSCTP handover management between two heterogeneous networks CDMA2000 and WLAN. Also, the metrics handover delay, end-to-end throughput and handover packet loss are chosen for determining whether the handover is seamless or not. We conduct a theoretical analysis for these three parameters in this chapter and we will compare them with the real handover delay, end-to-end throughput and handover packet loss from our testbed in chapter 5.

4.1 Mobile SCTP handover procedure between CDMA2000 and WLAN

Assume that a mobile node initiates an SCTP association with the correspondent node. For example, the MN downloads a file from the CN using SCTP as the transport layer protocol. The association between the MN and the server is through the CDMA2000 network. Then the MN starts to move across a WLAN, the scenario being depicted in following Figure 4.1:

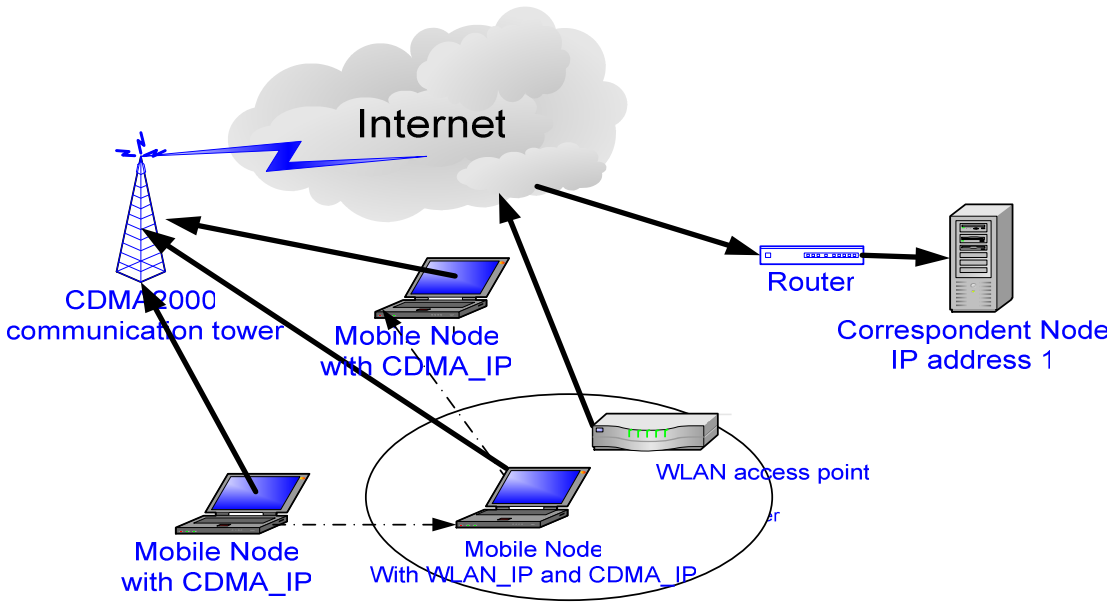


Figure 4.1 Handover between CDMA2000 and WLAN

Also assume the MN has two interfaces, one a CDMA2000 modem and the other a WiFi card. Thus, when the MN move into the WLAN, the WiFi card can obtain an IP address from the access point of the WLAN and the MN can use this IP address to communicate with the server. As the speed of WLAN is much higher and the cost of a WLAN is much lower, the MN prefers communicating via the WiFi card when it moves into the WLAN area and it only swaps back to the CDMA2000 modem when it moves away from the WLAN. Accordingly, there are two handovers: one is the handover from CDMA2000 to WLAN when the WiFi card becomes active; the other is the handover from WLAN to CDMA2000 when the WiFi card becomes inactive.

The following steps are performed for the handover from CDMA2000 to WLAN:

Step1: Obtaining an IP address for the new location. As the MN just moves into the WLAN area, it obtains a new IP address (say WLAN_IP) from the access point of the WLAN with the help of DHCP or IPv6 auto-configuration and becomes multi-homing.

Step2: Adding the WLAN_IP to the SCTP association. When the MN becomes

multi-homing, it needs to inform the CN of the new IP address by sending an ASCONF chunk with Add-IP parameter. In reply, the CN will return an ASCONF-ACK chunk.

Step3: Changing the primary IP address. As mentioned before, WLAN_IP is much faster and cheaper address for communication compared to the IP address of CDMA2000 (say CDMA_IP). Thus, the MN should inform the CN to change the primary IP address by sending an ASCONF chunk with Set-primary-IP parameter. As a reply, the CN will return an ASCONF-ACK chunk.

By now, the handover from CDMA2000 to WLAN is completed. A key difference between the handover from CDMA2000 to WLAN and the handover between two WLANs is that the last step of the handover between two WLANs is to delete the original IP address and make the MN single-homing again while this step is not performed for handover from CDMA2000 to WLAN. This is because connectivity to the CDMA2000 network exists at all times. When the primary IP address is changed to the WLAN_IP, the CDMA_IP is still active and it becomes the secondary IP address, which means that the MN remains multi-homed.

Before the MN leaves the WLAN area, it uses the WLAN_IP as the primary IP address for the association with the CN. Since the MN is still multi-homed, the MN does need to add CDMA_IP to the existing association and it can start the handover by merely changing the primary IP address. The following steps are performed when the MN leaves the WLAN area and moves back to CDMA2000 area:

Step1: Changing the primary IP address. The MN is still multi-homed so it does need to inform the CN to add a new IP address. When the MN detects that the WLAN_IP is becoming inactive, it informs the CN to change the primary IP address by sending an ASCONF chunk with Set-primary-IP parameter. In reply, the CN will return an ASCONF-ACK chunk.

Step2: Delete old IP address. After the MN leaves the WLAN area, the WLAN_IP becomes inactive but it is still in the association between the MN and the CN. Thus, deleting IP address is necessary in this case. To do so, the MN sends an ASCONF chunk with Delete-IP parameter, and in reply, the CN will return an ASCONF-ACK.

Thus to perform the whole handover using mSCTP, the MN exchanges four pairs of ASCONF/ASCONF-ACK control chunks with the CN. These signaling exchanges affect the performance of the handover. Figure 4.2 shows the mSCTP handover signaling messages.

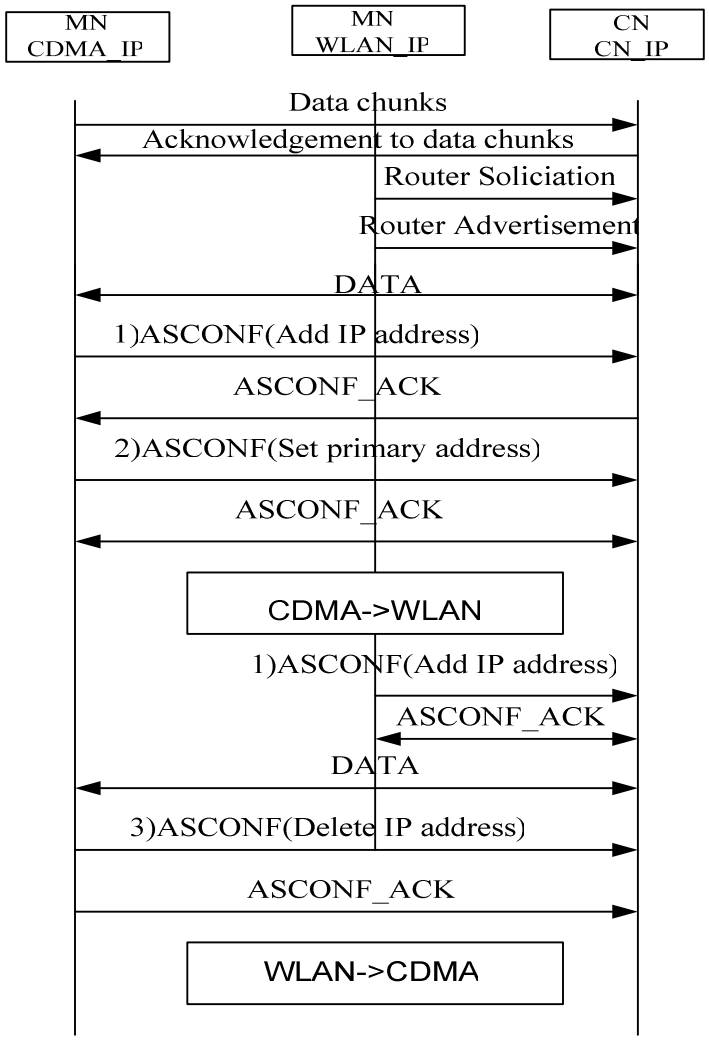


Figure 4.2 mSCTP handover signalling between CDMA2000 and WLAN

In Figure 4.2, two pairs of ASCONF/ASCONF-ACK control chunks exchange before the handover from CDMA2000 to WLAN occurs.

4.2 Handover delay

Handover delay is the most significant measurement for the handover. It can affect other handover quality metrics like packet loss and the end-to-end throughput. Handover delay is defined as the period of time between two given moments.

The first one is the moment when the MN move to a new subnet and the current primary IP address becomes inaccessible or less preferred for data transmission.

The second one is the moment when the endpoint switches the primary IP address to the new IP address the MN obtains from the new subnet.

More specifically, for our project, the handover delay when a MN moves into a WLAN is the period of time between the moment when the MN receives the last packet using CDMA_IP and the moment when the MN receives the first packet using WLAN_IP. On the other hand, the handover delay when a MN moves away from the WLAN is the period of time between the moment when the MN receives the last packet using WLAN_IP and the moment when the MN receives the first packet using CDMA_IP.

According to the mSCTP handover procedure presented in last subsection, mSCTP handover from CDMA2000 to WLAN generates not only the DAR procedure between the MN to the CN but also the router discovery procedure performed between the MN and the AP of the WLAN. In contrast, the handover from WLAN to CDMA2000 only introduces the DAR procedure performed between the MN and the CN. Now define some parameters for the analysis of handover delay, see table 4.1:

Parameter	**Definition**
T_{rs}	mSCTP router solicitation time
T_{ra}	mSCTP router advertisement time
$T_{new\text{-}IP}$	mSCTP new-IP processing time
$T_{add\text{-}IP}$	mSCTP add-IP time
$T_{del\text{-}IP}$	mSCTP delete-IP time
$T_{set\text{-}primary\text{-}IP}$	mSCTP set-primary-IP time
T_{ASCONF}	mSCTP ASCONF chunk send delay
$T_{ASCONF\text{-}ACK}$	mSCTP ASCONF-ACK chunk send delay
T_{rd}	Router discovery Period
T_{DAR1}	DAR procedure period from CDMA2000 to WLAN
T_{mSCTP1}	mSCTP handover delay from CDMA2000 to WLAN
T_{DAR2}	DAR procedure period from WLAN to CDMA2000
$T_{mSCTP12}$	mSCTP handover delay from WLAN to CDMA2000

Table 4.1 Definition of handover delay parameter

The following equations represent the handover delay when the MN moves across to the WLAN area:

$T_{mSCTP1} = T_{DAR1} + T_{rd}$
$T_{mSCTP12} = T_{DAR2}$

Router discovery (T_{rd}) consists of router solicitation time (T_{rs}), router advertisement (T_{ra}), and processing time to obtain a new IP address. That is:

$T_{rd} = T_{rs} + T_{ra} + T_{new-IP}$

The DAR procedure period is composed of the ASCONF parameters sent from the MN to CN to inform the changes. Thus

$T_{DAR1} = T_{add-IP} + T_{set-primary-IP}$

$T_{DAR2} = T_{del-IP} + T_{set-primary-IP}$

While

$T_{add-IP} = T_{ASCONF} + T_{ASCONF-ACK}$

$T_{set-primary-IP} = T_{ASCONF} + T_{ASCONF-ACK}$

$T_{del-IP} = T_{ASCONF} + T_{ASCONF-ACK}$

Consequently, mSCTP handover delay may be expressed as:

$$T_{mSCTP1} = (T_{rs} + T_{ra} + T_{new-IP}) + (T_{add-IP} + T_{set-primary-IP})$$
$$= (T_{rs} + T_{ra} + T_{new-IP}) + 2*(T_{ASCONF} + T_{ASCONF-ACK})$$

$$T_{mSCTP2} = (T_{del-IP} + T_{set-primary-IP})$$
$$= 2*(T_{ASCONF} + T_{ASCONF-ACK})$$

For the handover from CDMA2000 to WLAN, the handover delay consists of two parts. One is the router discovery time; the other is RTT (round trip time) of the ASCONF chunk between the MN and the CN. Theoretically, the router discovery time can be ignored due to the multi-home feature of SCTP. SCTP uses the WiFi card for router discovery while data are still being transmitted via the CDMA2000 card. Hence, the handover delay heavily depends on the RTT of 2 ASCONF chunks. For the handover from WLAN to CDMA2000, the handover delay is the RTT of 2 ASCONF chunk. Overall, the mSCTP handover delay is RTT of 4 ASCONF chunks between the MN and the CN this is why mSCTP handover is considered as a seamless handover.

4.3 Handover Throughput

In the previous section, we analyzed the handover delay of mSCTP. The second metric we analyse is the total end-to-end transmission throughput between the MN and the CN. The MN-to-CN transmission throughput is defined as the total number of data bits the MN receives during communication, including the handover delay duration.

Before we calculate the MN-to-CN transmission throughput, we define some parameters in Table 4.2:

Parameters	**Definition**
V_{msctp1}	MN-to-CN transmission throughput for handover from CDMA2000 to WLAN
V_{msctp2}	MN-to-CN transmission throughput for handover from WLAN to CDMA2000
T_{mSCTP1}	mSCTP handover delay from CDMA2000 to WLAN
$T_{mSCTP12}$	mSCTP handover delay from WLAN to CDMA2000
V_{DAR}	DAR chunk (ASCONF and ASCONF-ACK) overhead
T_s	Total duration time
P_{msctp1}	Total SCTP traffic for handover from CDMA2000 to WLAN
P_{msctp2}	Total SCTP traffic for handover from WLAN to CDMA2000
D_{msctp1}	Average data loss due to handover delay from CDMA2000 to WLAN
D_{msctp2}	Average data loss due to handover delay from WLAN to CDMA2000

Table 4.2 Definition of handover MN-to-CN transmission throughput parameter

The following equations represent the handover delay when the MN moves across the WLAN area:

$V_{msctp1} = [P_{msctp1} - (D_{msctp1} + 2 * V_{DAR})] / Ts$

$V_{msctp2} = [P_{msctp2} - (D_{msctp2} + 2 * V_{DAR})] / Ts$

V_{DAR} is overhead bits of ASCONF and ASCONF-ACK, where the overhead of ASCONF is 192 bits and ASCONF-ACK is 64bits when no error occurs to change the status of association.

VDAR= 192 + 64 = 256 bits

Lmsctp1 is average data loss due to handover delay from CDMA2000 to WLAN.

$D_{msctp1} = P_{msctp1} * (T_{mSCTP1} / T_s)$

$D_{msctp2} = P_{msctp2} * (T_{mSCTP2} / T_s)$

To sum up,

$V_{msctp1} = [P_{msctp1} - (P_{msctp1} * (T_{mSCTP1} / T_s) + 512)] / T_s$

$V_{msctp2} = [P_{msctp2} - (P_{msctp2} * (T_{mSCTP2} / T_s) + 2 * V_{DAR})] / T_s$

As shown in the above equations, the MN-to-CN transmission throughput depends on the handover delay as well. When the handover is seamless, the handover delay is almost zero and the MN-to-CN transmission throughput is almost the same as the throughput without handover. On the other hand, when the handover delay is high, it will reduce the MN-to-CN transmission throughput significantly.

4.4 Handover Packet Losses

In this section we discuss another handover measurement—pocket loss. Packet loss is the total number of lost pockets during handover epochs. It also depends on the handover delay. Since we can make the packet size fixed, say S, the packet losses can be defined as the total data loss divided by the packet size. In the previous section, we already defined the data loss during handover, so the packet loss for handover can be

expressed as follows:

$$Smsctp1 = \frac{Pmsctp1 * (T_{mSCTP1}/Ts)}{S}$$

$$Smsctp2 = \frac{Pmsctp2 * (T_{mSCTP2}/Ts)}{S}$$

Where S_{msctp1} is the packet loss for handover from CDMA2000 to WLAN and S_{msctp2} is the packet loss for handover from WLAN to CDMA2000.

From this equation, we can draw the same conclusion to the MN-to-CN transmission throughput. When the handover is seamless, the handover delay is almost zero and almost no packets are lost during handover, whereas when the handover delay is high, many packets are lost during handover. The retransmission of these packets cause network congestion and lowers the service quality.

Chapter 5
Testbed and Software Development

In this chapter we first describe how to set up our handover testbed between CDMA2000 and WLAN networks, which includes how to configure mSCTP as well as the CDMA2000 and WLAN interfaces on a Linux platform. Then the handover software is presented, including the core ideas and the software requirements.

5.1 Testbed development

Our testbed consists of two computers. A desktop running Fedora core3 with kernel version 2.6.9 is the server. A laptop running Fedora core3 with the same kernel version is the client and the test mobile node. The laptop uses two interface adapters to connect the server: one is a PCMCIA GTran dotSurfer CDMA2000 wireless modem provided by Telecom New Zealand, the other one is a Dlink USB WiFi card. An access point is installed in the office to provide WiFi connectivity to the Dlink USB card. Figure 5.1 shows the appearance of the testbed.

There are three main tasks in setting up the testbed. The first one is to implement mSCTP on both the client and the server. SCTP can be used in many OS (operation system) platforms such as FreeBSD, WindowsXP and Linux. However, I concentrate on Linux because Linux can support mSCTP and there is much freedom to change the source code of the Linux kernel. Specifically, Redhat Linux with kernel 2.6.9 or higher can be used for the platform with some special Linux patches. We present the Linux settings in section 5.1.1.

Figure 5.1 The Testbed

The second task is to set up the hardware properly. For the CDMA2000 Gtran modem, we need modify the Windows driver into a Linux driver. For the WiFi USB card, some software called Ndiewrapper was downloaded to convert its Windows driver into a Linux driver. Hardware setting is described in section 5.1.2.

The third task is to build a HTTP over SCTP server. Apache is chosen to be the HTTP server but it uses TCP/UDP as its transport protocols. Thus, some modifications need to be made in order to use SCTP as the transport protocol between the client and server. Building the HTTP server is presented is section 5.1.3.

5.1. 1 Linux Setting

Before we start to set up SCTP in Linux, Redhat Linux 9.0 with kernel over 2.6.9 should be prepared. In fact, we used Redhat Linux 9.0 with kernel version 2.6.9 and Fedora core3. Because Fedora core 3 has SCTP kernel as the kernel module, kernel recompile is not needed for our case. Instead, the SCTP kernel module is simply needed to be loaded into the RAM memory on the Fedora core 3 with the command 'modprobe'. The command 'modprobe SCTP' plays a role on loading SCTP module into the RAM; see Figure 5.2

Figure 5.2 Load SCTP module into the kernel

The SCTP module should be loaded on both the server and the client. After that, it can be assumed that both the server and client have already configured theLinux platform so that they are capable of supporting the SCTP protocol. The next step is to activate the DAR extension of SCTP, to ensure that mSCTP is supported by Linux. The parameter 'addip_enable' is the indicator whether DAR extension is active or not. When 'addip-enable' is 0, Add-IP extension is inactive while it is active when 'addip-enable' is 1. Command 'echo 1>/proc/sys/net/sctp/addip_enable' is used to make Linux support mSCTP. Command 'more /proc/sys/net/sctp/addip_enable' approves the information. See Figure 5.3:

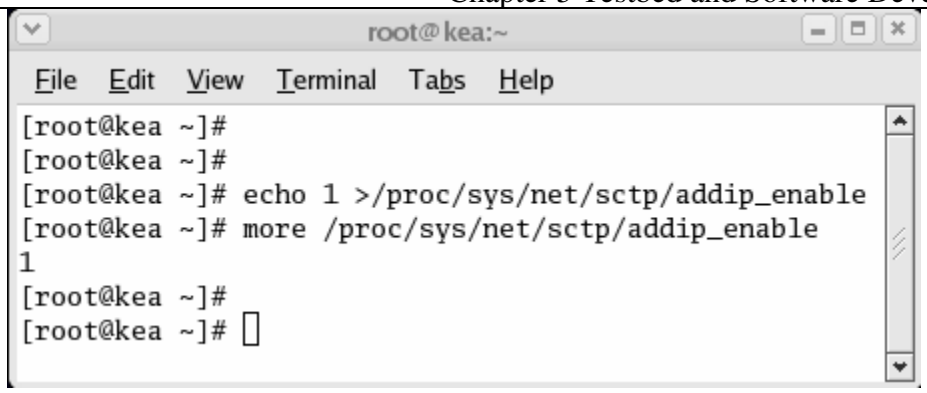


Figure 5.3 Active Add-IP extension of SCTP

One problem with the SCTP protocol in Linux is that it does not support SCTP APIs itself, while SCTP APIs are required to be used for coding the mSCTP handover between CDMA2000 and WLAN networks. At this point, we downloaded an additional tool from http://sourceforge.net/project/showfiles.php?group_id=26529 called LKSCTP [25], which is able to provide SCTP API functions. There are many versions of the LKSCTP tool, the latest one is 1.0.6. The one used in our testbed is version 1.0.5. The following steps have been taken to build LKSCTP in Linux:

1. Become root user to install LKSCTP by command: su –
2. Enter the LKSCTP directory containing the download RPM files of LKSCTP by command cd /root/fde13 (my own directory).
3. Install the RPM flies by command: rpm *.lksctp-tools-1.0.5-1.rpm
4. Untar the LKSCTP tools directory from the gzipped tarball by command: tar –xzvf lksctp-tools-1.0.5.tar
5. Enter the LKSCTP tool directory by command: cd /lksctp-tools-1.0.5
6. configure LKSCTP by command: ./configure
7. make LKSCTP by command: make

After the success of “make” operation, the LKSCTP tools has been loaded into the Linux kernel. The following Figure shows how to check whether LKSCTP is supported by Linux or not.

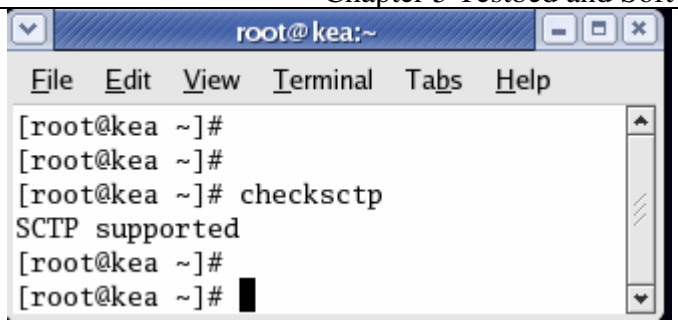


Figure 5.4 LKSCTP tools for Linux

In Figure 5.4, the command '**checksctp**' indicates whether the server and the client support LKSCTP or not. The result shows that both of them support LKSCTP.

5.1.2 Hardware Settings

Linux setting is reasonably straight forward, just following the instructions on the website, but setting up the devices requires some tricks. As mentioned before, the client is being run on an Acer laptop with Sempron 2800+ CPU and it is equipped with a PCMCIA CDMA2000 Gtran modem and a Dlink USB WiFi card.

5.1.2.1 CDMA2000 Gtran Modem

Getting the CDMA2000 Gtran card to run has nothing to do with modem dialer programs. Although a username, password and phone number was supplied by Telecom New Zealand with the card, a connection cannot be established by just inputting all these information to a dialer program. Instead, **KPPP** is required for launching the Gtran card. The following steps are taken to install the drivers:

1. Become the root user to start the **KPPP** program.
2. Input the username and password, phone number, modem name (gtran) device name (ttyS3), line termination (CR) connection speed (57600).
3. Enter to configure the commands for the Gtran card

4. Change the “initialization string 1” to “ATZAT&C0” which means setting the CF high all the time.
5. Change the “initialization string 2” to “AT+IFC=2,2 X4” which means setting the hardware flow control for TX/RX and turning on better result code.

Now the CDMA2000 Gtran Card has been successfully installed in Linux and it can be used to connect to the Internet. After launching the KPPP program and putting the username and password in, the CDMA2000 Gtran card can connect to the Internet, see Figure 5.5.

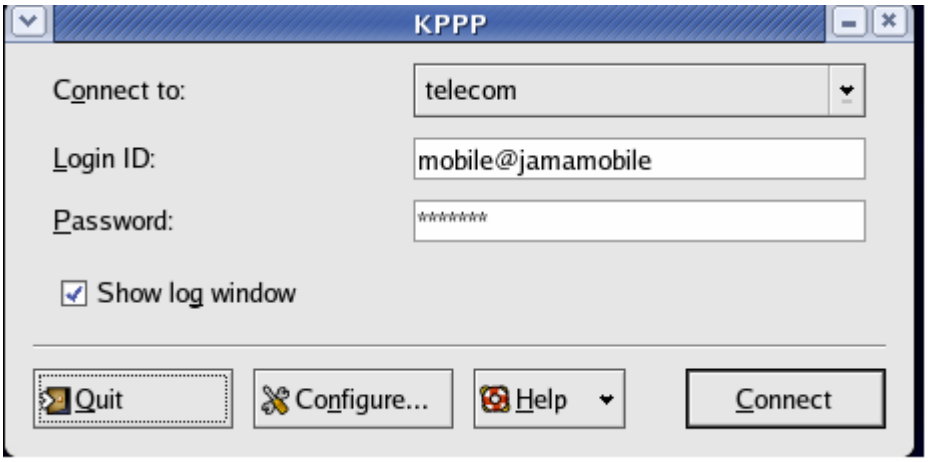


Figure 5.5 Launch the CDMA2000 Gtran card

Although the Gtran card can be used to connect to the Internet, there have been some problems with the Gtran card:

1. The CDMA2000 Gtran card can not connect to the internet indoors sometimes because the signal strength is not strong enough. Moving the laptop close to the window makes connection to the Internet easier.
2. Sometimes it takes quite a long time to obtain the connection. The normal connection time should be less than 30 seconds while the average connection times was found to be about 3 minutes. The reasons why sometimes the CDMA2000 Gtran card cannot obtain the connection are still uncertain. Changing

the value of CDMA2000 Gttran card timeout and Pre-init delay may help to reduce the connection obtaining time.

3. Sometimes the connection is alive but the traffic can not go through. This is because the signal strength is not very stable and the connection can not be used for traffic when the signal strength is not high enough.

After launching the CDMA2000 Gtran card successfully, the KPPP statistics show the details of the connection (see Figure 5.6). In this Figure, of importance is the local address which is the CDMA_IP address used for handover procedure and this IP address differs every time the card connects to the Internet. The bytes in and out indicate whether traffic can go through the card or not.

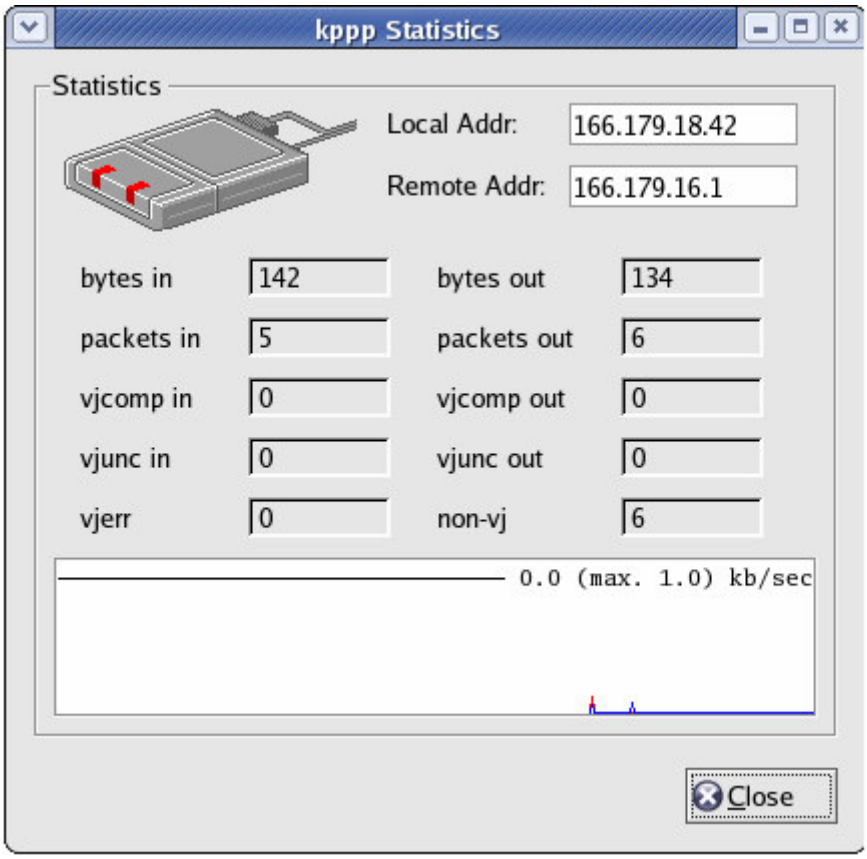


Figure 5.6 Statistics of CDMA2000 connection

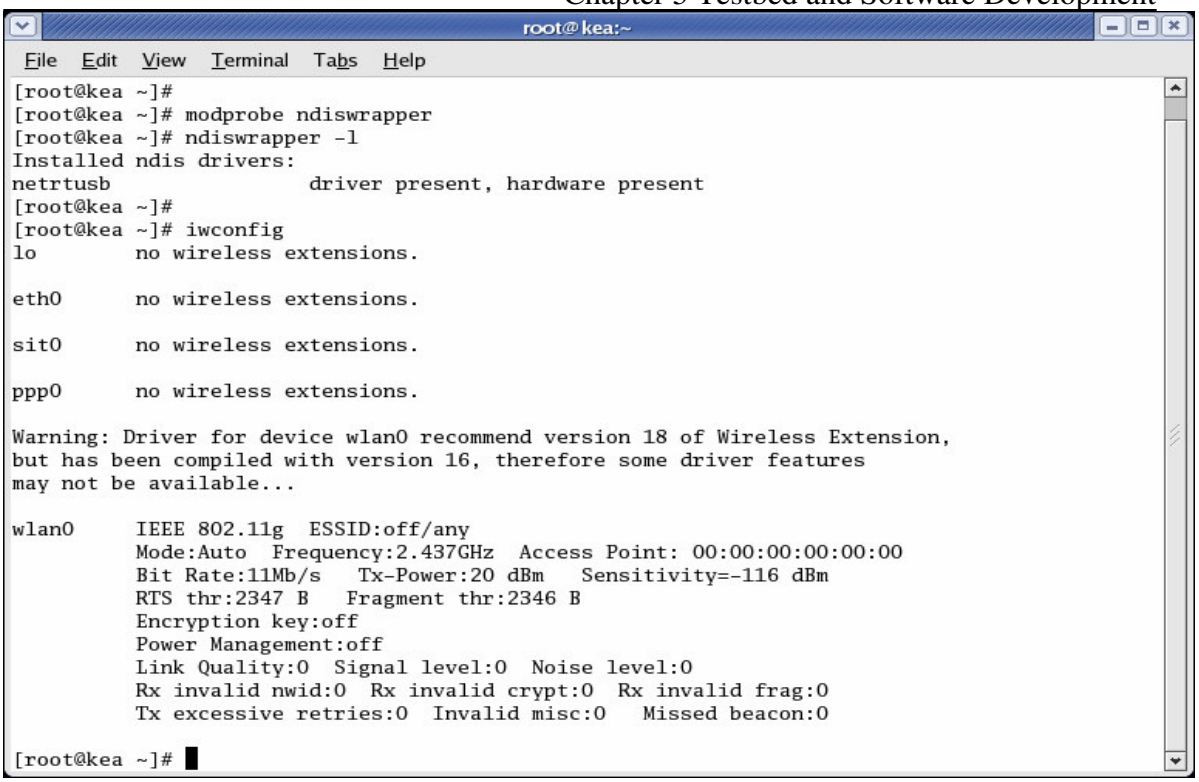
5.1.2.2 Dlink USB WiFi Card

In this subsection we describe how to install the Dlink USB WiFi card under Linux and how to set up the card. Although several wireless drivers are built into the Linux kernel, none of them supports the USB WiFi card. To make the WiFi card work, an additional piece of software called ndiswrapper [37] is required to convert the Windows driver into a Linux drivera. Before downloading and installing ndiswrapper, one has to make sure that the kernel version is at least 2.6.6 and there is a link to the kernel sources from the modules directory. After that, do the following steps to install the Dlink USB WiFi card:

1. Download ndiwwrapper with version 1.5 from http://sourceforge.net/
2. Become root user to install the driver by command: su-
3. Extract ndiswrapper by command: `tar -zxvf ndiswrapper-1.5.tar.gz`
4. Enter the ndiswrapper directory and run command: make, make install.
   Now the ndiswrapper can work in Linux successfully.
5. Load the ndiswrapper module into Linux kernel by command: modprobe ndiswrapper.
6. Insert the CD with Windows driver of Dlink USB WiFi card and change to CD directory.
7. Convert the Windows driver to Linux driver with command: ndiswrapper –i netrtusb.inf

After converting the Windows driver to Linux driver, the WiFi card has been installed in Linux successfully. The most parameter of WiFi card is the signal strength which is the measurement to start the handover. The command “iwconfig” is used to check the information of the WiFi card.

Figure 5.7 shows whether the WiFi card works properly or not and all the detail information of a WiFi card including the signal strength.



.

Figure 5.7 WiFi card in Linux

In Figure 5.7, the command '**ndiswrapper -l**' indicates whether the WiFi card is present in Linux or not. The result shows "driver present hardware present" and it is regarded as interface "wlan0" by Linux. Now command "iwconfig" is used to obtain the detail of WiFi card. The results show that the WiFi card can not get an "ESSID" and "IP address" itself and the WiFi card needs to be set up. Doing the following steps configures the card:

1. Find all the access points by command: iwlist wlan0 scan. After this command, all the access points come out. In my case, two access points come out, one is "UCwireless" and the other is "eleceng-r218" which is the access point I set up.
2. Choose the access point I set up by command: iwconfig essid 'eleceng-r218'.
3. In order to use this access point, the web key has to be set up by command: iwconfig wlan0 restricted key XXXXXXXXXX
4. The WiFi card obtains an IP address by command: iwconfig 132.181.52.138 netmask 255.255.255.0 broadcast 132.181.255.255.

Now the Wifi card has been set up completely and the command shows all the detail of the WiFi card in Figure 5.8:

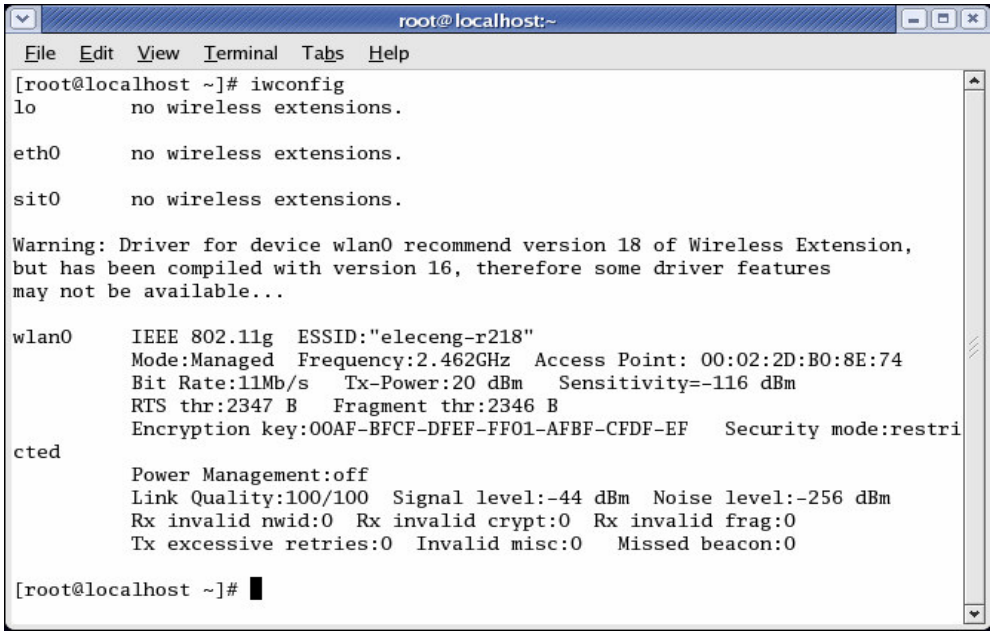


Figure 5.8 Information of WiFi card in Linux

In Figure 5.8, of significance is the signal level of the WiFi card. In this case, it is -44dBm, it is very strong at the moment because the client is close to the access point. When the client moves around, the signal level will change accordingly. This is fundamental to the handover procedure, in other words, the signal level of the WiFi card should be detected all the time.

5.1.3 Server setting

We installed LKSCTP in section 5.1.There are two components of LKSCTP tools playing important roles in our tesebed. One is that the run-time library of LKSCTP which can support SCTP API. The other one is “withsctp” tool, which uses SCTP to replace TCP for existing TCP binaries. Based on “withsctp tool”, we modified a TCP Apache server into a TCP-style SCTP Apache server.

The version of the TCP Apache server we used is 2.0.54 and it was downloaded from the website http://www.apache.org/dist/httpd/httpd-2.0.54.tar.gz. The following steps have been done to install the TCP Apache server and change it into a TCP-style SCTP server:

1. Extract the Apache server by command: `tar -zxvf httpd-2.0.54.tar.gz`
2. Enter into the Apache server directory by command : cd httpd-2.0.54
3. Run the configure file by command: ./configure
4. Edit the listen file of the server by command: nedit server/listen.c
5. Go to the line 289 of listen.c and change the following two lines from

   if ((status = apr_socket_create(&new->sd,

   SOCK_STREAM, process->pool))

   to

   if ((status = apr_socket_create_ex(&new->sd,

   SOCK_STREAM, IPPROTO_SCTP, process->pool))

   Then the change has been made to open a TCP-style SCTP socket instead of a TCP socket.
6. Save the file and exit.
7. Run the install files by command: make install.

   The TCP Apache server has been installed at /usr/local/apache2
8. If neccessary, the port number can be changed so that the server can be run by an ordinary user, by the port number 80, to say 8080. In order to do this, enter the Apache server directory by command: cd /usr/local/apache2
9. Edit the httpd.conf file by command: nedit httpd.conf
10. Go to line 220, change the port number from 80 to 8080.
11. Start the server by command: /usr/local/bin/apachectl –k start

By now, the TCP-style SCTP server has been installed and the testbed has been set up completely. The configuration of each component is according to Figure 5.9:

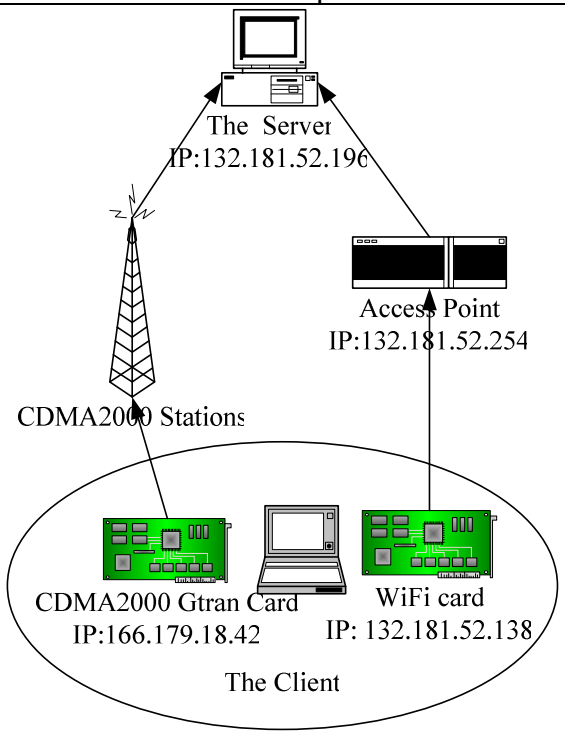


Figure 5.9: Configure of the testbed

The client can download data from the web server either through the CDMA2000 Gtran card or the USB WiFi card by the command:

withsctp wget **http://132.181.52.196:8080/** , see Figure 5.10

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# withsctp wget http://132.181.52.196:8080/
--22:33:28--  http://132.181.52.196:8080/
           => `index.html.1'
Connecting to 132.181.52.196:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12,885 [text/html]

100%[====================================>] 12,885        --.--K/s

22:33:28 (223.27 KB/s) - `index.html.1' saved [12,885/12,885]

[root@localhost ~]#
```

Figure 5.10: SCTP association between the MN and the CN

In Figure 5.10, the client uses SCTP to download the index file from the server by CDMA2000 or WiFi. The only difference between CDMA2000 and WiFi is the speed; WiFi is about 100 times faster than CDMA2000. However, the "index.html" file is too small, it only takes a few seconds to complete the SCTP data transmission. In order to perform handover during the SCTP association, we made a large file called "test_large_file.ext" to increase the SCTP data transmission time by the following command:

cat /dev/zero >test_large_file.ext

chmod a+r test_large_file.ext

The "test_large_file.ext" is all zero and it is 284.5 MB. The location of the "test_large_file.ext" is the same as the location of "index.html", which is /usr/local/apache2/htdocs

The client can download this file by command:

withsctp wget http://132.181.52.196:8080/test_large_file.ext

Both the CDMA2000 Gtran card and the USB WiFi card need long time to download the file. Hence the handover between CDMA2000 and WLAN can be performed during the downloading of this file.

5.2 Software development

After setting up the testbed, the next step is to develop the software to perform the handover between CDMA2000 and WLAN. Language C is used to be programming tool for the software. Furthermore, three application programming interfaces (APIs) are required to be used to perform the handover [26]:

1. sctp_bindX(ADD): The function of this API is to add the WLAN IP to the SCTP association when the MN detects that it moves into to a WLAN area and obtain an IP address from the AP.

2. sctp_bindx(REMOVE): The function of this API is to delete the WLAN IP from the SCTP association when the MN moves far away from a WLAN and it can not get signal from the AP.

3. setsockpt(PRIMARY_PEER_ADDR): The function of this API is to change the primary IP address from CDMA_IP to WLAN_IP when the MN detects the signal strength of WiFi card is strong enough and change the primary IP address from WLAN_IP and CDMA_IP when the MN detects the signal strength of WiFi is not strong enough.

In addition, these SCTP APIs functions are written on Unix Network Programming. In order to use those functions, sources and library available from http://www.unpbook.com/unpv13e.tar.gz are required to download.

Figure 5.11 shows how these SCTP APIs functions are used to perform mSCTP handover. Four time slots have been set up in figure 5.11 to use these functions. When the MN moves into the WLAN and SCTP performs the ADD_IP operation by calling the function sctp_bindx(add) at 'Time 1'. Then the MN keeps moving close to the AP, SCTP performs the primary IP change operation at 'Time 2'. At 'Time 3', the MN starts to moves away from the WLAN area and SCTP performs primary IP change again. Thus, function setsockopt(PRIMARY_PEER_ADDR) are calling at 'Time 2' and 'Time 3'. At 'Time 4', SCTP calls function sctp_bindx (remove) to delete the WLAN_IP which is already down.

The signal strength of IwiFi card is the measurement to decide when to change the status of each IP address. [27][28][29]. That is to say, the values of the four time slots are controlled by the signal strength of WiFi card. In order to use the signal strength of WiFi card for handover, four thresholds are needed to set up to decide the values of four time slots, see table 5.1:

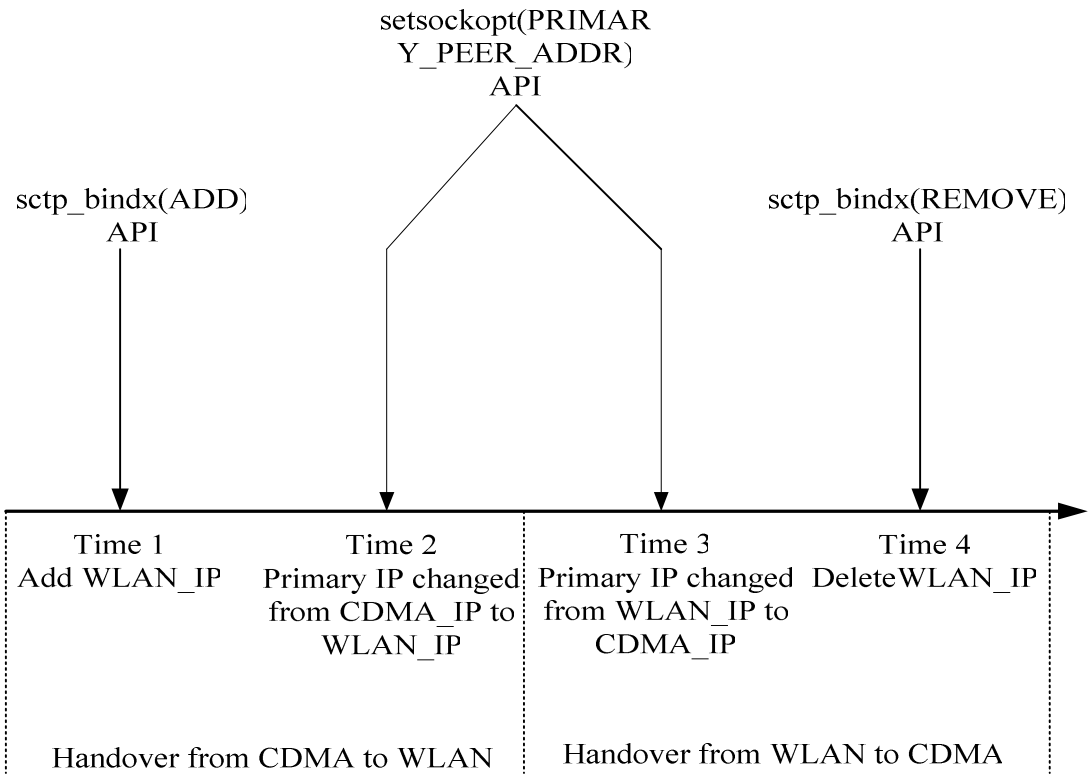


Figure 5.11 Handover Procedures and used APIs

Thresholds_add_IP	Signal strength value when the add_ip procedure can be performed by SCTP
Thresholds_set_primary_IP_WLAN	Signal strength value when the primary IP address can be changed from CDMA_IP to WLAN_IP by SCTP
Thresholds_set-primary-IP-CDMA	Signal strength value when the primary IP address can be changed from WLAN_IP to CDMA_IP by SCTP
Thresholds_delete_IP	Signal strength value when the delete_IP procedure can be performed by SCTP

Table 5.1 Signal strength thresholds

In table 5.1, thresholds_add_ip is used to add the WLAN_IP to the existing SCTP association when the signal strength reaches to some specify value, say T1, which defines the values of 'Time1' in figure 5.11. On the other hand, when the signal strength gets weaker and reduced to T4, which means the WiFi signal is not strong enough to make the WLAN_IP be the secondary IP address for the existing SCTP association. Thus, the delete_ip procedure is performed when the signal strength is weaker than T4, which defines the value of 'Time 4'.From this point of view, thresholds_add_ip equals to thresholds_delete_IP are the values to decide whether WiFi IP can be the secondary IP address for the assocaiton. They should equal each other: T1=T4, sat T1 for both of them.

Thresholds_set_primary_IP_WLAN is used to change the primary IP address from CDMA_IP to WLAN_IP when the signal strength reaches to some specify values, say T2, which defines 'Time 2'. On the other hand, when the signal strength gets weaker and reduced to T2, which means the WiFi signal is not strong enough. Thus, the primary IP address should be changed from the WLAN_IP to CDMA_IP when the signal strength is weaker than T3, which defines 'Time 3'. From this point of view thresholds_set_primary_IP_WLAN equals thresholds_set-primary-IP-CDMA, say T2 for both of them.

As I mentioned before, command "iwconfig" displays all the information of the WiFi card, including the signal strength. In our project, we use command "iwconfig" to measure the signal strength of the WiFi card and write the value into a file. The MN read the file to obtain the signal strength and decide how to perform the handover.

After defining specified value T1 and T2 for these four thresholds, the flow control of the software can be seen in figure 5.12 according to the handover procedure describe in the previous chapter.

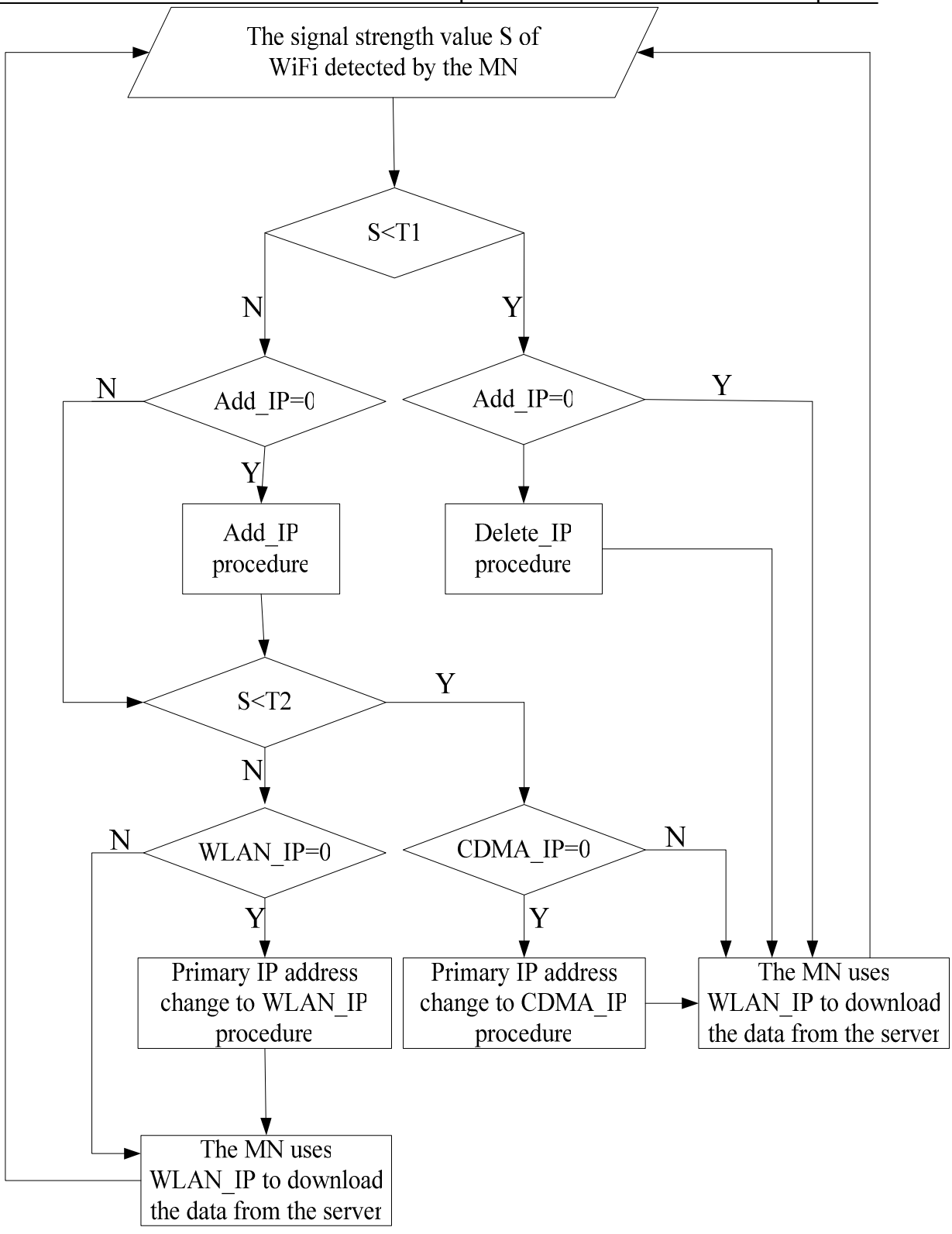


Figure 5.12 Flow control according to signal strength

The parameters in figures are explained in tables 5.2:

parameter	Value	Description
ADD_IP	0	Only one IP address (CDMA_IP)in the existing SCTP association
WLAN_IP	0	WLAN_IP is the secondary IP address
CDMA_IP	0	CDMA_IP is the secondary IP address.
ADD_IP	1	CDMA_IP and WLAN_IP are both working
WLAN_IP	1	WLAN_IP is the primary IP address
CDMA_IP	1	CDMA_IP is the primary IP address.

Table 5.2 Status parameters of IP address in SCTP association.

Now we implement the following pseudo-code based on the flow control:

1: Set threshold for WLAN:

T1= -82db T2=-65db

2: Connect to CDMA2000.

CDMA_IP=1 WLAN_IP=0

3: While 1 do

4: If -65db > signal strength of WLAN > -82db then

ADD_IP=1

CDMA_IP=1

WLAN_IP=0

The association becomes multi-homing.

5: End if.

6: If signal strength of WLAN > -65db

ADD_IP=1

CDMA_IP=0

WLAN_IP=1

The connection has been changed from CDMA to WLAN.

7: End if.

8: If -65db > signal strength of WLAN > -82db then

ADD_IP=1

CDMA_IP=0

WLAN_IP=1

The connection has been changed back from WLAN to CDMA.

9: End if.

10. If signal strength of WLAN < -82db

ADD_IP=0

CDMA_IP=1

WLAN_IP=0

The association becomes single-homing again.

12: End if.

13: End while.

In this chapter we present how we set up the testbed and the basic idea to perform mSCTP handover in our testbed. We will show the performance of mSCTP handover in our testbed in next chapter.

Chapter 6

Results and Discussion

We have presented the development of our handover testbed and its software in the last chapter. In this chapter, we focus on the performance of mSCTP handover on the testbed. Handover delay, end-to-end throughput and packet loss are the major parameters for mSCTP handovers.

Total handover delay: In order to obtain a handover delay from the testbed. We use ethereal [30] to capture all the packets coming into or going out from the server.
The handover delay is measured by the time difference between the time a handover it triggered and the time a data packet is received successfully after the handover period.
End-to-end throughput: From the trace of all the packets for the server, we account the amount of packets transmitted by TSN.
Packet loss: The packet loss is measured by counting the number of packets loss during the handover period.

We started the SCTP association using CDMA2000 modem to download the test file from the server. The signaling messages play the roles as the indication to change the status of the association. Figure 6.1 shows the signaling messages presented the whole process.
As depicted in figure 6.1, the RTT time of Add_IP process is 0.000032s while it is 0.000013s and 0.000022s for Set_primary_IP process and Delete_IP process respectively. Thus, these process times are too short and they can be ignored compared to the handover delay.

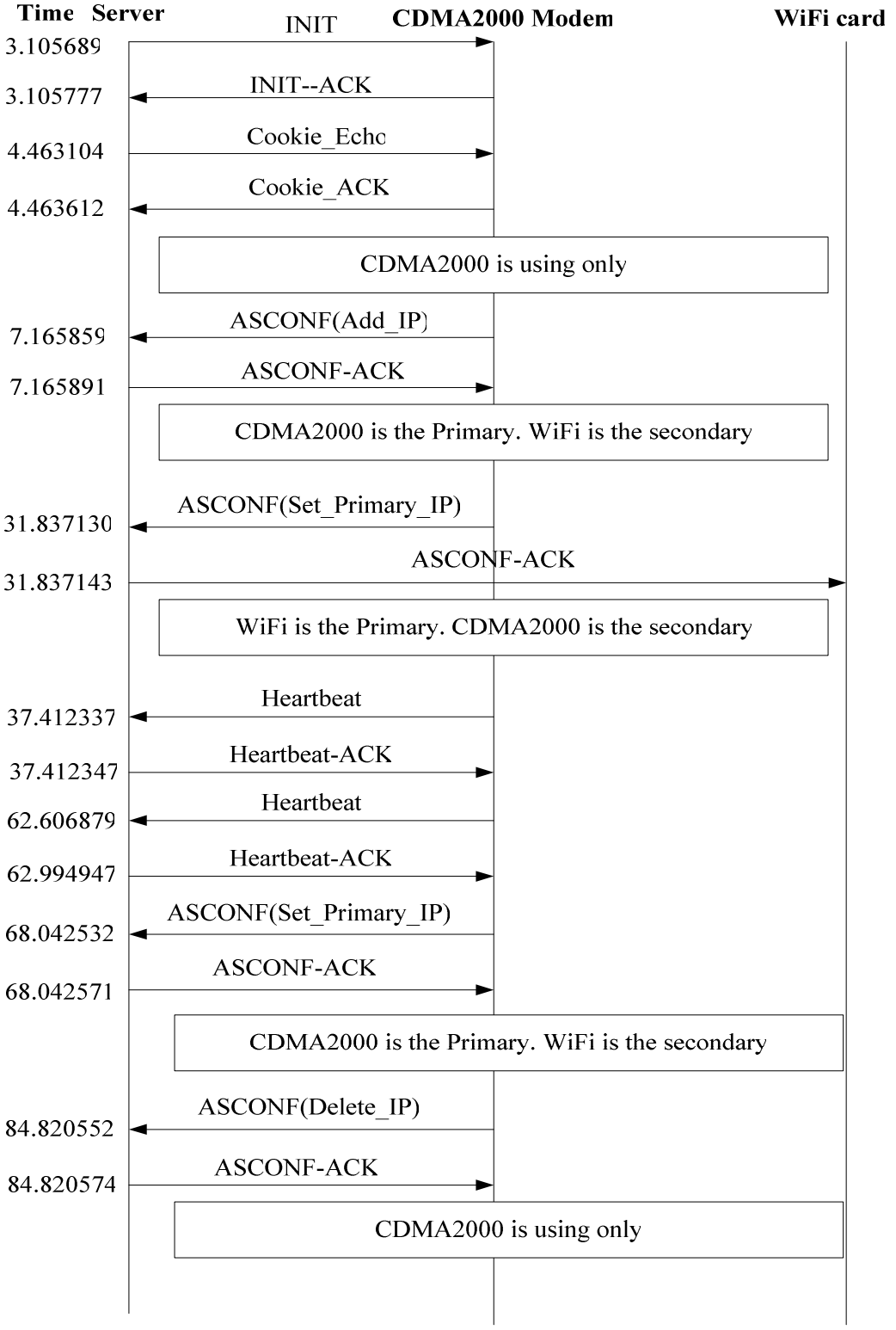


Figure 6.1 Signaling diagram of the handover process

6.1 Handover Delay

Handover occurs after the client and the server exchange the message of Set-Primary-IP. In this subsection we present the handover delay between CDMA2000 and WLAN networks.

It is very easy to see the handover delay from CDMA2000 to WLAN in figure 6.2. The last packet using CDMA2000 as the primary IP address before the handover is at time 31.834756 and the first packet using WiFi as the primary IP address after the handover is at time 31.837153. Therefore, the handover delay is 31.837153 - 31.834756 = 0.002397s. Comparing handover delay to the RTT time of Set_Primary_IP, we can see that the RTT time is not the main components of handover delay while how to measure signal strengths is more critical to reduce handover delay.

Next, we observe the handover delay from WLAN to CDMA2000 in figure 6.3. Before The last packet using WiFi before handover is at time 66.521750. However, this packet is lost. We will analyze the reasons why this packet can not be received by the client in next subsection. Then, the secondary IP address- CDMA2000 is used to retransmit this packet. This is why a packet using CDMA2000 for the transmission before the handover. In fact, the first packet using CDMA2000 after handover is at time 68.521586. Therefore, the handover delay is 68.521586-66.521750=1.999836s.

We can see that the handover delay from WLAN to CDMA2000 is heavily larger than the handover delay from CDMA2000 to WLAN. There are three main causes to make handover from WLAN to CDMA2000 huge. One is the retransmission time, which is 67.521809-66.521750=1.000059s. The second is that the response time to retransmitting message by the server, which is 68.042532-67.521809=0.520723s. The third is the delay time when the client receives the ASCONF-ACK message and it starts to use the CDMA2000 as the primary IP address, which is 68.521586-68.042571=0.479015s. All these might be due to the signal strength of CDMA2000 which is not strong enough in

our office. Figure 6.4 shows the packets using WiFi for retransmission when CDMA2000 IP address is the primary IP address.

There are 21 retransmitting packets between time 7.165891 when WiFi starts as the secondary IP address and time 31.837143 when WiFi starts as the primary IP address, which means about 0.86 packet is required to be retransmitted in one second when CDMA2000 is used for the connection between the client and server. This also happens when the primary IP address changes from WiFi to CDMA2000. 7 pakcets are required to retransmit between time 68.042571 when WiFi becomes secondary IP address and time 84.820574 when WiFi is deleted from the association, which shows about 0.41 packet is required to be retransmitted in one second. This can be seen in figure 6.5. All these retransmissions show that the signal strength of CDMA2000 in our office is not good or the the CDMA2000 modem used for our testbed does not work perfectly.

To sum up, the handover from CDMA2000 to WLAN using mSCTP is very stable and low latency. As a result, it is seamless. On the contrary, the handover from WLAN to CDMA2000 produces high latency because of the strength of CDMA2000.

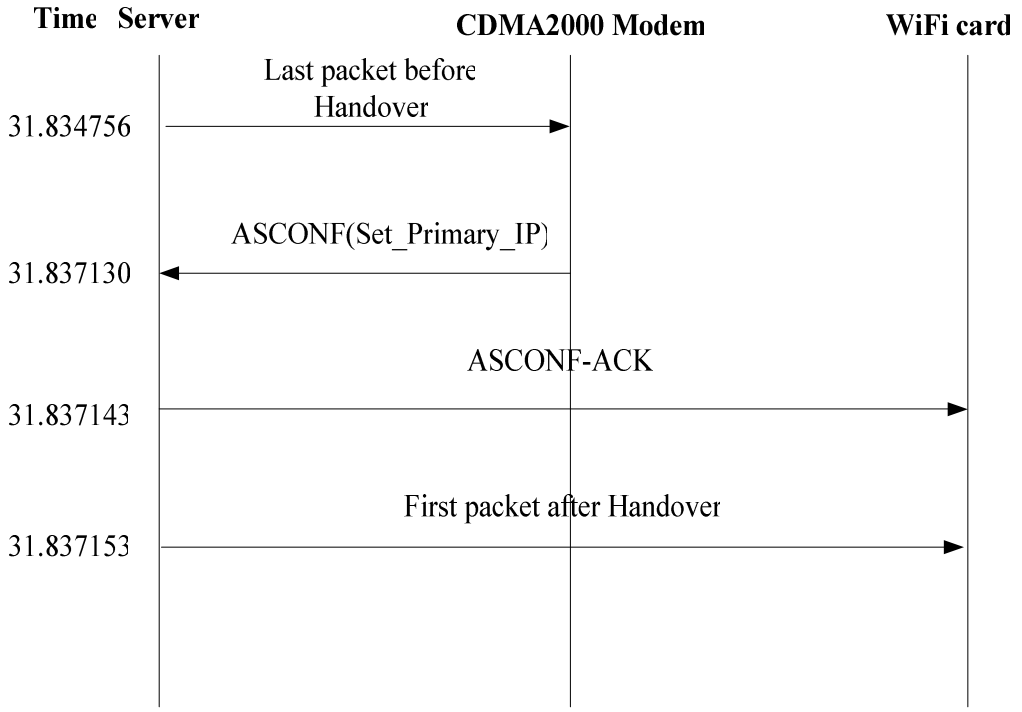


Figure 6.2 Handover delay from CDMA2000 to WLAN

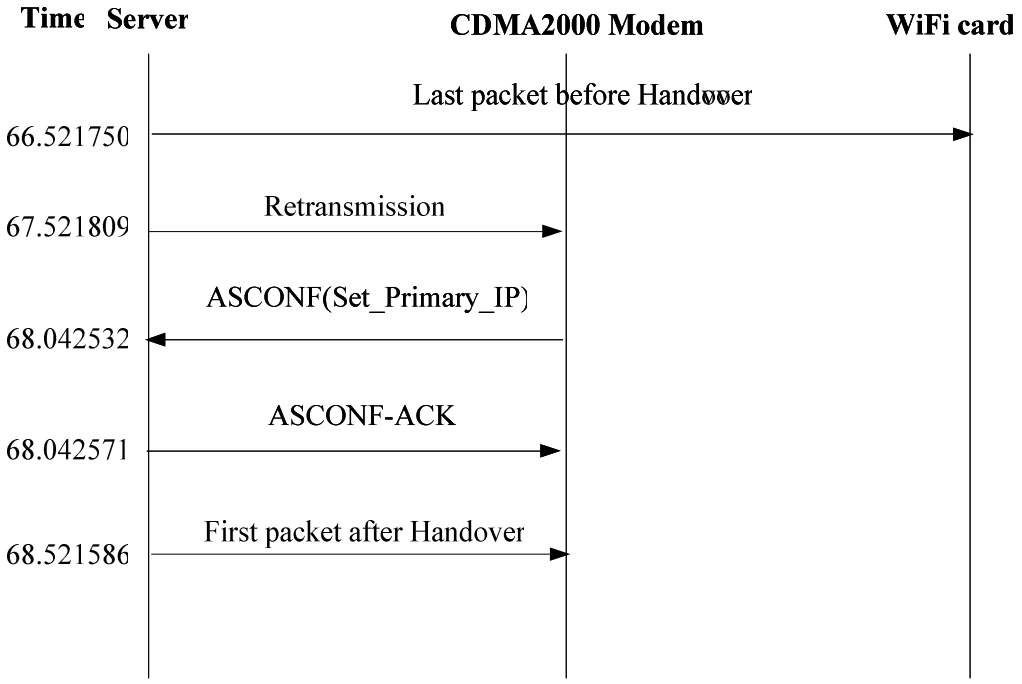


Figure 6.3 Handover delay from WLAN to CDMA2000

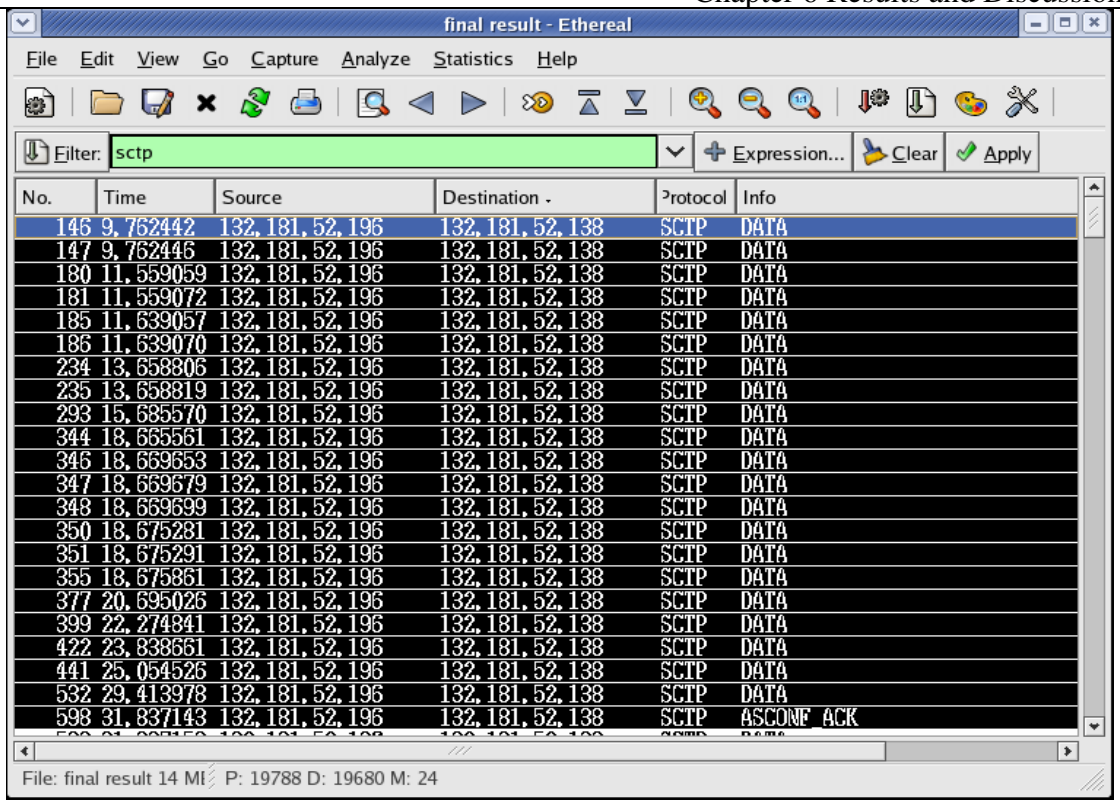


Figure 6.4 Retransmission when CDMA2000 is primary IP address

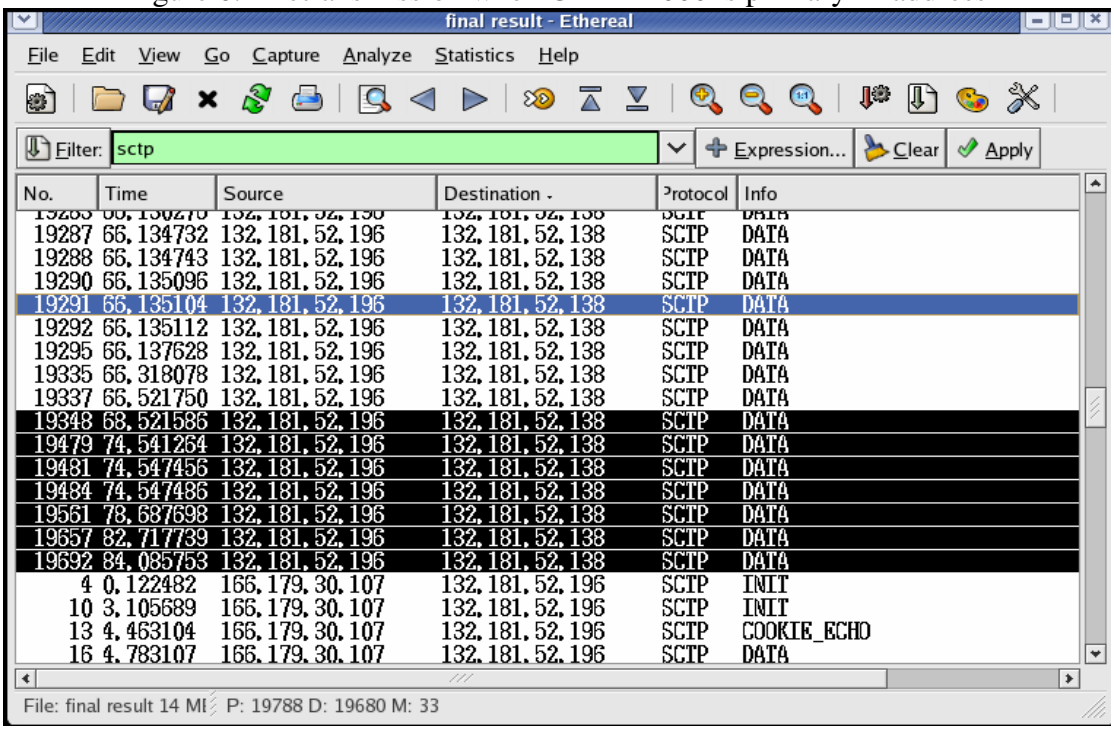


Figure 6.5 Retransmission when CDMA2000 is primary IP address

6.2 End-to-End Throughput

In this subsection we present the end-to-end throughput by measuring the packets at the server. Figure 6.6 illustrates the throughput for the whole process.

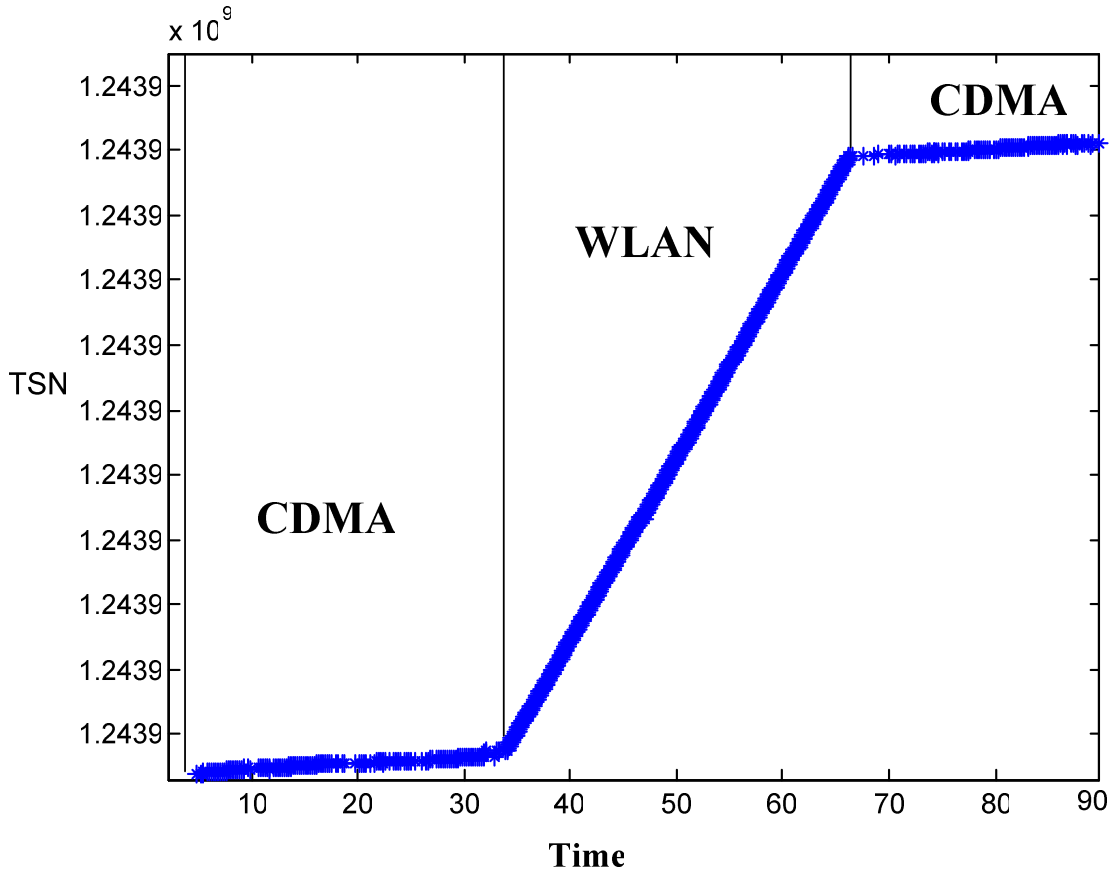


Figure 6.6 Handover throughputs against time

It can be seen that as the data rate of CDMA2000 is very low compared to WLAN, a significant increase of throughput can be observed for handover from CDMA2000 to WLAN occurs while a sharp reduction is generated for handover from WLAN to CDMA2000. More importantly, the throughput does not go to zero during the handover.

6.3 Packet Loss

In this subsection we present the packet loss during handover by measuring the packets at the server. Figure 6.7 shows all the SACK packets received by the server with the DATA packets.

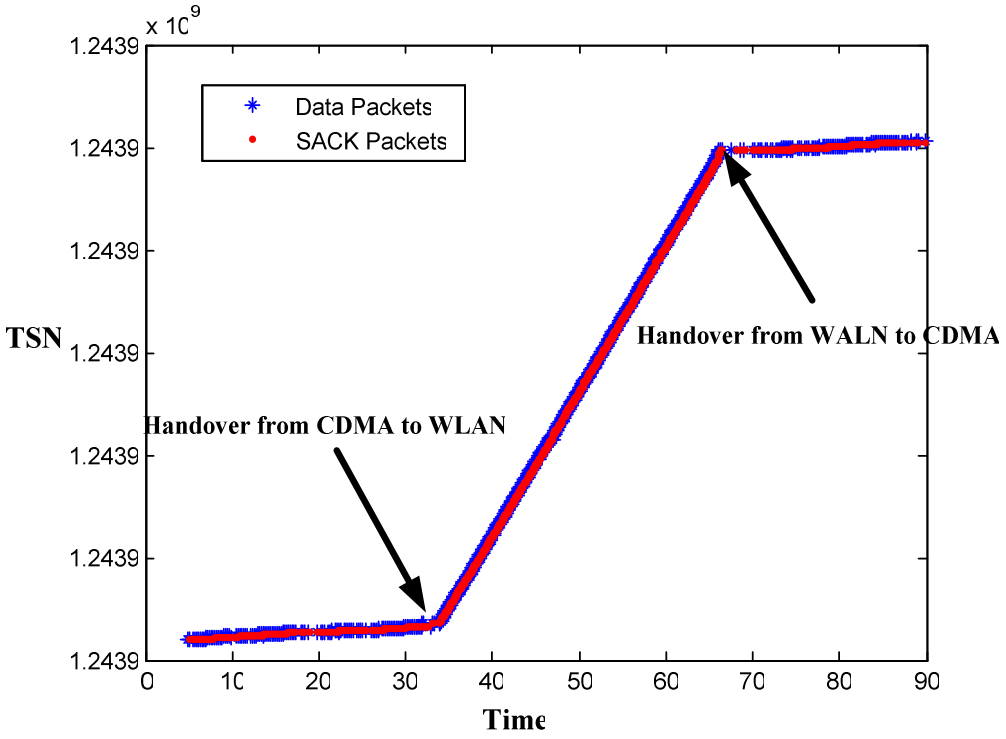


Figure 6.7 SACK packets and Data Packets against time

For the handover from CDMA2000 to WLAN, the TSN of the last data packet is 1243934652 while the last sack packet received by the server before handover shows that the client only receives the packets up to 12439346661, which indicates that the data packets with TSN between 1243934661 and 1243934652 are still on the way. These ten packets use the CDMA2000 IP address as the primary IP address. After the handover, one packet is lost which is retransmitted by the WiFi at time 33.868333. That is to say, the packet loss is one during the handover. The SACk packets show that all other packets are received by the client with CDMA IP address.

For the handover from WLAN to CDMA2000, only one packet with TSN 1243943932 is still one way which uses WiFi as the primary IP address. The SACK packet shows that it is received at time 70.935885. Thus, the packet loss is zero during the handover.

6.4 Discussion and Future Work

In this subsection we will present the problems existing in our mSCTP handover between CDMA2000 and WLAN.

The handover performance shows that mSCTP handover is soft and smooth for handover from CDMA to WLAN. The handover delay is only 2 mini-second, which can be almost ignored. However, there is one packet loss during the handover, which introduce a time gap for the transmission. Figure 6.8 shows the time gap.

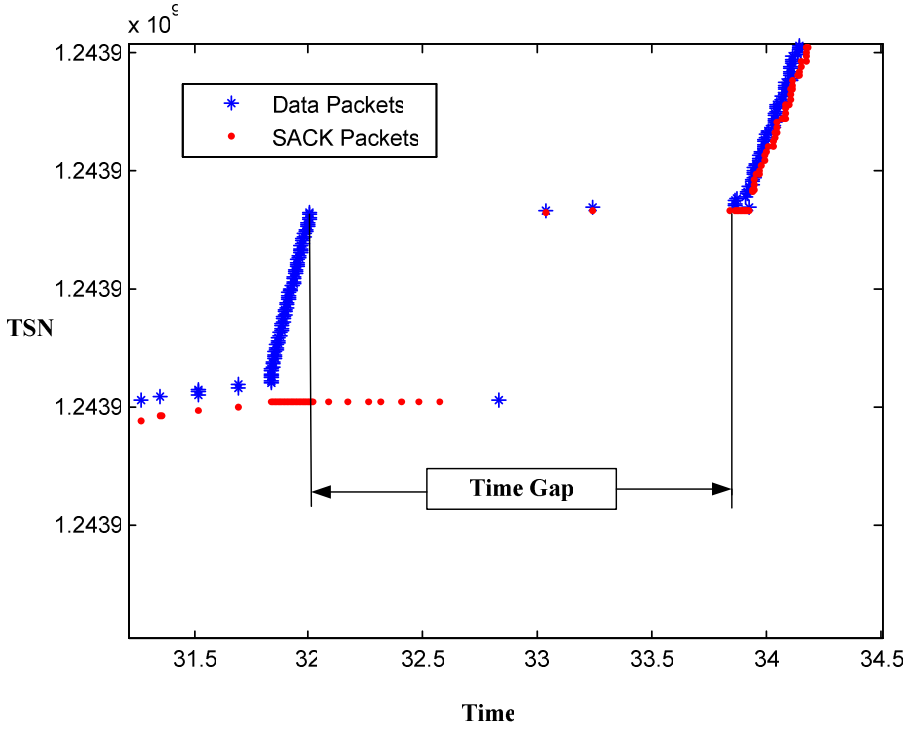


Figure 6.8 A time gap during transmission after handover

As I mentioned in last subsection, 10 packets are still on the transmission when the primary IP address is changed from CDMA to WiFi. These packets are using CDMA as the primary IP address and they are transmitted slower than the data packets using WiFi. Thus, the SACK packets show that the client does not receive these packets (see the red dot in figure 6.8).

We can use the size of RWND to explain the results. The size of RWND is displayed in the detailed information of the SCTP packet. It can be seen that the size of RWND is still very large so the server can use WiFi to transmit the data packets. When these packets are received by the client, the size of RWND is reduced. Figure 6.9 is the size of RWND for the whole process.

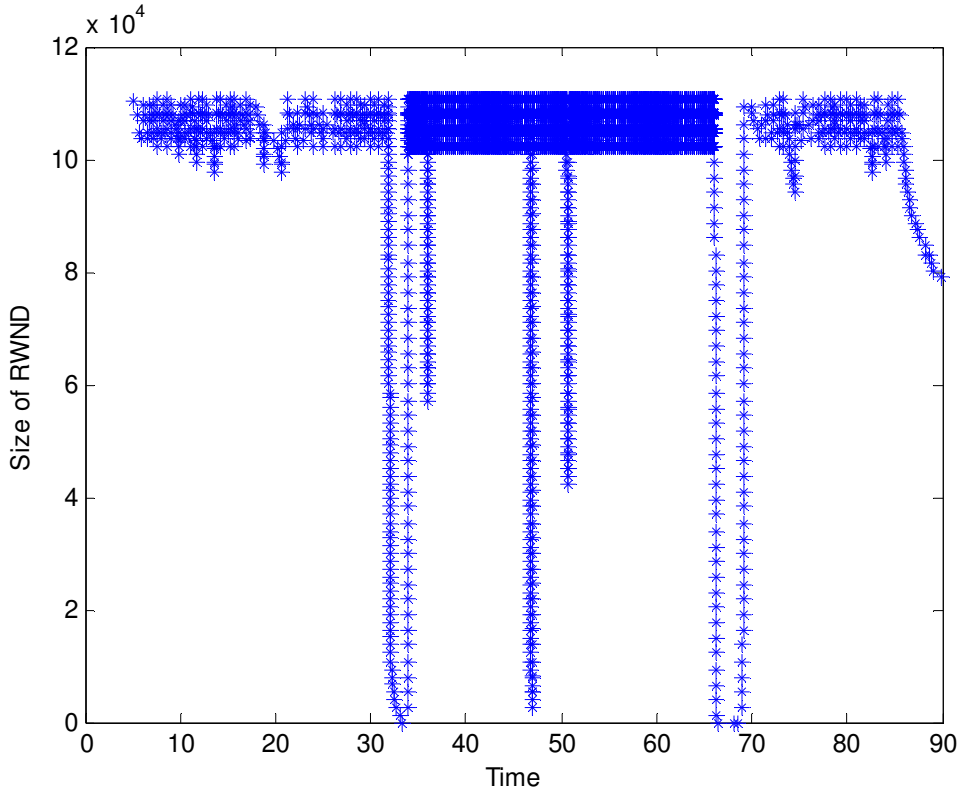


Figure 6.9 Size of RWND against time

From Figure 6.9, we can see that the size of RWND is reduces to zero during the time gap.

For SCTP, "the sender must not transmit any new data to a given transport address of the receiver if there are already cwnd or more bytes of user data to that transport address outstanding". [13] That is to say, when the size of RWND is the total of the packets transmitted by the WiFi after handover and the 10 packets which are still on the way by the CDMA, the server can not send any new packets to the client. This is why no packets are transmitted until the loss packet is retransmitted by the WiFi.

After the retransmission of the loss packet and one packet from the WiFi, the size of RWND is zero already. For SCTP, "the data sender must not transmit new data to the receiver if the rwnd of the receiver indicates that there is no receive buffer space left (that is, the rwnd equals zero). However, regardless of the value of rwnd (including if it is zero), the data sender can always have one Data chunk in flight to the receiver if it is allowed by the cwnd". [13]. Thus, the server sends one more data packet to the receiver. As a result, there are only three packets on transmission during the time gap, which decreases the throughput.

From figure 6.9, same problem occurs at time 66.521708 and this is the main cause to make the delay for handover from WLAN to CDMA huge. Figure 6.10 shows this time gap.

To sum up, mSCTP handover is a low-loss and low-latency end-to-end mobility management. Here we have introduced mSCTP handover to a heterogeneous network and it still has some problems which express as a time gap during the transmission time. The main cause of the time gap might be the signal strength of CDMA for our testbed is not good enough. Because SCTP is a relatively new protocol, some immature aspects of SCTP might negative affect to the handover as well. The future work is to find the real reason for the time gap and reduce it.

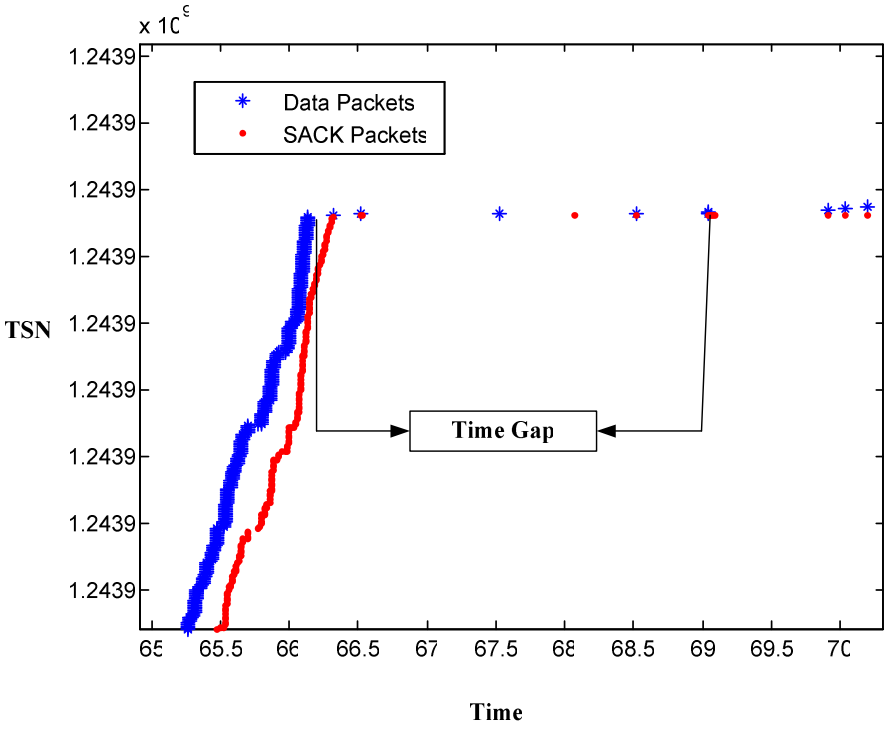


Figure 6.10 Time Gap for handover from WLAN to CDMA

References

[1] C. Perkins, “IP mobility support”, IETF RFC 3344, August 2002.

[2] Koh, S., Lee, M., Riegel, M., Ma, M. and M. Tuexen, "Mobile SCTP for Transport Layer Mobility", Internet draft, draftsjkoh-sctp-mobility-03.txt, February, 2004.

[3] http://en.wikipedia.org/wiki/OSI_model

[4] R. Ramjee, T. La Porta, S. Thuel, and K. Varadhan, “IP Micro-Mobility support through HAWAII”, IETF Internet Draft, draft-ramjee-micro-mobility-hawaii-01.txt, July 2000.

[5] M.Ratola “Which layer for mobility? Comparing HIP, MIPv6 and SCTP.” Helsinki University, April, 2004.

[6] A.G.Valko, “ Cellulary IP : A New Approach to Internet Host Mobilty” Computer and Communication, Review, Vol.29, no.1, Jan, 1999.

[7] Pablo Brennel, “A technical Tutorial on IEEE802.11 protocol”, 1997 http://www.sss-mag.com/pdf/802_11tut.pdf

[8] Gast, M. S., “802.11 Wireless Networks – The Definitive Guide”, 1st edition, 2002.

[9] Crow, B. P. et al.: “IEEE 802.11 Wireless Local Area Networks”, IEEE Communications Magazine, 1997, vol35, issue 9, p116-126.

[10] Christopher Wingert, and Mullaguru Naidu, "White Paper: CDMA1×RTT Security Overiew", August 2002.

[11] "an overview of IS-95 and CDMA2000", July 2002 http://www.geocities.com/rahulscdmapage/Technical/Chapter4.pdf

[12] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream control transmission protocol" IETF RFC 2960,2000

[13] Randall. R. Stewart and Qiaobing Xie. Stream control transmission protocol (sctp): A reference guide. Addison Wesley, 2002.

[14] Shaojian Fu and Mohammed Atiquzzaman, "Sctp: State of the art in research, products, , and technical challenges", IEEE Communications Magazine, 42(4):64–76,April 2004.

[15] Ong, L. et al.: "An Introduction to the Stream Control Transmission Protocol (SCTP)", RFC 3286, 2002.

[16] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Stream Control transmission protocol (sctp) dynamic address reconfiguration", May 2006 http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-addip-sctp-15.txt

[17] Perkins, C.E., "Mobile Networking Through Mobile IP", IEEE Internet Computing, 1998, vol. 2, issue 1, p. 58-69.

[18] S.Koh, M.J.Chang, and M.Lee "mSCTP for soft handover in transport layer" IEEE Communications letters March 2004.

[19] D.Johnson, “Mobility support in IPv6” RFC3775, 2004.

[20] J. Rosenberg “SIP:Session Initiation Protocol” RFC 3261, 2002.

[21] H.Schulzrinne, and E.Wedlund, “Mobility Support using SIP” ACM/IEEE International Conference on Wireless and Mobile Multimedia, 1999.

[22] Thomas Ravier, Rob Brennan and Thomas Curran, “Experimental studies of SCTP multi-homing” Teltec DCU, Dublin 9, Ireland.

[23] S. Fu, Atiquaazman, “ Trash: A Transport Layer Seamless Handover for Mobile Networks” University of Oklahoma 2004.

[24] A. Festag, H. Karl, and G. Sch¨ afer “Current developments and trends in handover design for ALL-IP wireless networks” Technical University Berlin, Aug, 2000.

[25] Linux Kernel SCTP Project, Available from http://lksctp.sourcforge.net/

[26] R. Stewart, Q. Xie, L. Yarroll, K. Poon, and M. Tuexen, \Sockets api extensions for stream control transmission protocol (SCTP)." IETF DRAFT, June 2006.

[27] Chang, M., et al., “Transport Layer Mobility Support Utilizing Link Signal Strength Information”, IEICE Transactions on Communications, September 2004.

[28] Ma. l. et al.: “A New Method to Support UMTS/WLAN Vertical Handover Using SCTP”, IEEE Vehicular Technology Conference, 2003.

[29] Dong Phil Kim, Jong Shik Ha, Sang Tae Kim and Seok Joo Koh “use of SCTP for IP handover” Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science, 2005.

[30] Ethereal, Available from http:// www.ethereal.com

[31] Nokia: "Introducing Mobile IPv6 in 2G and 3G mobile networks", Nokia, 2001.

[32] G. Montenegro "Reverse Tunneling for Mobile IP" RFC2344 May 1998
.

[33] H. Schulzrinne, and E.Wedlund, "Application-Layer Mobility Using SIP" IEEE Service Portability and Virtual Customer Environment, 2001.

[34] H.Soliman, "Mobile IPv6: Mobility in a Wireless Internet" 2004.

[35] R. , and M.Chris, "SCTP: New Transport Protocol for TCP/IP" IEEE Internet Computing, Nov, 2001.

[36] C.Perkins, David B. Johnson "Route Optimization in Mobile IP" 2001.

[37] Ndiswrapper, Available from http:// www.ndiswrapper.com

Appendix: Part Source Code to implement mSCTP handover on our testbed

The headfile client.h

```
/* all the headfiles used for our code*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <netdb.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <sys/uio.h>
#include <netinet/in.h>
#include <sys/errno.h>
#include <errno.h>
#include <netinet/sctp.h>
#include <wait.h>
#include <signal.h>

#define BUFSIZE 8192    //The size for the buffer
static char buffer[BUFSIZE];
```

```
static int buffer_size = BUFSIZE;
static int serv_port = 8080;    //The port number for HTTP service

static void emsg();
void print_notification();   //Print the notification of the association
void sctp_print_addresses();  //Print the information of the address of the association
```

The execute file client.c

```
int main(int argc, char *argv[])
{
        int ret, sock, addr_len, msg_flags;
      struct sockaddr_in clnt_addr, serv_addr; // define the structure of IP address for both
                                                the client and server
      struct sctp_sndrcvinfo sinfo, info;

/* intermediate data structures */
        struct sctp_assoc_change *sac;
        struct sockaddr_storage *sal, *sar, *sadel, *saprim;
        int num_rem, num_loc, len, iter;
        struct sockaddr_in add_addr, del_addr, total_addr[2], *rem_addr;
        extern int errno;

      /* create the socket*/
sock = socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);
        if(sock < 0)     emsg("socket");

/* set the socket option to event */
        struct sctp_event_subscribe event;
        int opt_len = sizeof(event); ;
```

```
        bzero(&event, opt_len);

        /* event notifications */
        event.sctp_data_io_event = 1;
        event.sctp_association_event = 1;
        event.sctp_address_event = 1;
        event.sctp_send_failure_event = 1;
        event.sctp_peer_error_event = 1;
        event.sctp_shutdown_event = 1;
        event.sctp_partial_delivery_event = 1;
        event.sctp_adaption_layer_event = 1;
        setsockopt(sock, IPPROTO_SCTP, SCTP_EVENTS, &event, opt_len);

/*set up the options for the server */
bzero(&serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
      serv_addr.sin_addr.s_addr = inet_addr("132.181.52.196");      // Specify the server
      address
        serv_addr.sin_port = htons(serv_port);                  // Server port = 8080


/*set up the options for the first IP address of the client (CDMA)*/
bzero(&clnt_addr, sizeof(clnt_addr));
        clnt_addr.sin_family = AF_INET;
       clnt_addr.sin_addr.s_addr = inet_addr("166.179.22.246"); //Specify the first IP
                                                                address
        clnt_addr.sin_port = htons(6767);                       // Client port = 6767

/* Set the secondary address of the client (WiFi)*/
        bzero(&add_addr, sizeof(add_addr));
```

```
        add_addr = clnt_addr;
    add_addr.sin_addr.s_addr = inet_addr("132.181.52.138"); //Specify the secondary IP
                                                                address
        add_addr.sin_port = htons(6767);        //Client Port =6767

/* Specify the first address and the secondary address*/
bzero(total_addr, sizeof(total_addr));
        total_addr[0] = clnt_addr;
        total_addr[1] = add_addr;

/* bind the first address */
 ret = sctp_bindx(sock, (struct sockaddr *)&clnt_addr, 1, SCTP_BINDX_ADD_ADDR);
        if(ret < 0) emsg("bindx");

        addr_len = sizeof(serv_addr);
        ret = connect(sock, (struct sockaddr *)&serv_addr, addr_len);

        if(ret < 0) {
                emsg("connect");
                printf("error\n");
                return;
                }
        printf("\n Client is now connected to Server ...\n");

/* Set the address features to the sock option*/
        bzero(&info, sizeof(info));
        info.sinfo_flags |= SCTP_ADDR_OVER;
        ret = setsockopt(sock, IPPROTO_SCTP, SCTP_DEFAULT_SEND_PARAM,
&info, sizeof(info));
        if(ret < 0) emsg("setsockopt-param");
```

```
/* Download a file from the server */
            bzero(buffer, BUFSIZE);
                  strcpy(buffer,"GET /test_large_file.ext HTTP/1.0\r\n\r\n");
                  len = strlen(buffer);



                  ret = sctp_sendmsg(sock, buffer, len, NULL, 0, 0, 0, 0, 0, 0);
                  if(ret < 0 ) emsg("sctp_sendmsg");
                  printf("\n Client received data messages to Server ! \n");

/* Start the client procedure */
          iter = 0;
          int num, addip,wifiip,cdmaip;
                  FILE *fp

/*initialize the status of the IP addresses*/
            addip = 0;
                  wifiip=0;
                  cdmaip=1;

while ( 1 )
          {
                  printf("\n\nIterations: %d \n", ++iter);

/* detect the signal strength of the WiFi card */
            fp = fopen("out1.txt", "r+");
              fseek(fp, 0L, SEEK_END);
              system ("iwconfig wlan0 | grep ' Signal level' | awk '{print $4}' >> out1.txt");
```

```
                fgets(buff, 100, fp);
                fclose(fp);

                num = atoi(&buff[6]); // the signal strength from command “iwconfig”
                printf("%d\n", num);

/* association between the server and the client */
   bzero(buffer, BUFSIZE);
                ret = sctp_recvmsg(sock, buffer, BUFSIZE,
                        (struct sockaddr *)&serv_addr, &addr_len, &sinfo, &msg_flags);
                if(ret < 0 ) emsg("sctp_recvmsg: association is closed !");

                if ( !(msg_flags & MSG_NOTIFICATION) )
                {
                        buffer[ret] = 0;
                        printf("\n Client received %d Bytes from Server", ret);
                                        fflush(stdout);
                }
                else
                {
                        printf("\n Notification Received **********************\n");
              union sctp_notification *sn;
                        sn = (union sctp_notification *)buffer;

                        printf("\n");
                        print_notification(sn);
                }

// ADD-IP Procudure
```

```
if ((num>-82)&&(addip == 0)&&(iter>2))   // When signal strength becomes the value
                                                    of T1
        {
                addip = 1;
                pid = fork();
                if(pid == 0)
                {
                        printf("\n enter into wlan area\n");
           exit(0);
                }
                wait(&status);
                printf("\n WiFi card is active now ! \n");

/* Bind the IP address to the WiFi card*/
                bzero(&add_addr, sizeof(add_addr));
                add_addr = total_addr[1];
                ret = sctp_bindx(sock, (struct sockaddr *)&add_addr, 1,
SCTP_BINDX_ADD_ADDR);
                if(ret < 0)        emsg("add-ip");


                printf("\n Now, a new address is added !!\n");

// Primary-Change procedure from CDMA2000 to WLAN
if ((num>-65)&&(iter>25)&&(cdmaip==1)) // When signal strength becomes the value
                                                    of T2

        {
      struct sctp_setpeerprim primary;
```

```
                        primary.sspp_assoc_id = 0;
                        primary.sspp_addr = *(struct sockaddr_storage *)&add_addr;
                        len = sizeof(primary);

                        ret = setsockopt(sock, IPPROTO_SCTP,
SCTP_SET_PEER_PRIMARY_ADDR, &primary, len);
                        if(ret < 0) emsg("peer-primary-change");

                        printf("\n The primary address is the address of the WiFi card
now !!\n");
        wifiip=1;
                        cdmaip=0;

                }

// Primary-Change
            if ((num<=-65)&&(cdmaip==0))  //When signal strength reduces to the value
                                                of T2
      {
                struct sctp_setpeerprim primary;

                        primary.sspp_assoc_id = 0;
                        primary.sspp_addr = *(struct sockaddr_storage *)&clnt_addr;
                        len = sizeof(primary);

                        ret = setsockopt(sock, IPPROTO_SCTP,
SCTP_SET_PEER_PRIMARY_ADDR, &primary, len);
                        if(ret < 0) emsg("peer-primary-change");
```

```
                    printf("\n Now, the primary address is the address of CDMA
card !!\n");
                    cdmaip=1;
                    wifiip=0;
                }

// Delete IP Procedure
                if ((num<-83)&&(addip=1))  //When signal strength reduces to the value of
                                                T1
                {
                 bzero(rem_addr, sizeof(rem_addr));

               rem_addr = &add_addr;
                    ret = sctp_bindx(sock, (struct sockaddr *)rem_addr, 1,
SCTP_BINDX_REM_ADDR);
                    if(ret < 0)        emsg("del-ip");
        addip=0;
               printf("\n Now, the address of the WiFi card was deleted !!\n");
                  }

/* Print Peer Addresses */
                  num_rem = sctp_getpaddrs(sock, NULL, (struct sockaddr **)&sar);
                  if(num_rem > 0){
                  printf("There are %d remote addresses: \n", num_rem);
                  sctp_print_addresses(sar, num_rem);
                  sctp_freepaddrs((struct sockaddr *)sar);
                  }
                  /* Print Local Addresses */
                  num_loc = sctp_getladdrs(sock, 0, (struct sockaddr **)&sal);
                  if(num_loc > 0){
```

```
                printf("* There are %d local addresses: \n", num_loc);
                sctp_print_addresses(sal, num_loc);
                sctp_freeladdrs((struct sockaddr *)sal);
                }

/*close the socket*/
if (iter==4000)
        {close(sock);
        exit(0);
```