ENSC835: HIGH-PERFORMANCE NETWORKS
CMPT 885: SPECIAL TOPICS: HIGH-PERFORMANCE NETWORKS

Final Report

Simulation of Handoff Procedure based on SIP over Wireless LAN

Fall 2003

Tao Jia
Qiang Hou
Wei (Eric) Peng
Email address: {tjia,qhou,wep}@sfu.ca
URL: http://www.sfu.ca/~tjia

TABLE OF CONTENT

1. Abstract **3**

2. Introduction **3**

2.1 Wireless LAN and IEEE 802.11 Standard **4**

2.2 Wireless LAN Mobility **4**

2.3 SIP overview **5**

2.4 Mobility Support using SIP over WLAN **7**

3. OPNET Models for Wireless LAN **9**

3.1 Node models **9**

3.2 Interested Statistics **10**

4. Project Implementation in OPNET **11**

4.1 Assumptions **11**

4.2 SIP models in OPNET **11**

5. Simulation Results and Problems still existing **16**

6. Discussion and conclusion **17**

7. Future work **18**

Reference: **19**

Appendix A: code listing **20**

Appendix B: H.323 protocols suite **24**

List of Figures and Tables

Fig.1 Ad Hoc network **Fig.2 Infrastructure BSS** **4**

Table.1 SIP Request method **5**

Table. 2 SIP Response code **6**

Fig.3 SIP Session Establishment and Call Termination **7**

Fig. 4 Pre-Call mobility: registration procedure **8**

Fig. 5 Mid-Call mobility: reINVITE procedure **8**

Fig. 5 Mid-Call mobility: reINVITE procedure **8**

Fig. 6 Typical timing for SIP-based handoff procedure in WLAN **9**

Fig. 7 WLAN workstation **Fig. 8 WLAN Access Point** **10**

Fig. 9 Caller side subnet **Fig. 10 Callee side subnet** **10**

Fig. 11 Hierarchy relationship within application layer **12**

Fig. 12 Network topology **12**

Fig. 13 Process model: gna_clsvr_mgr **13**

Fig. 14 Process model: gna_profile_mgr **13**

Fig. 15 Modified process model: gna_voice_calling_mgr **14**

Fig. 16 Process model: sip_UAC_mgr **14**

Fig. 17 Modified process model of sip_UAC **15**

Fig. 18 Process model: sip_UAS **16**

Fig. 19 Voice application configuration **16**

Fig. 20 Average ETE delay (no handoff) **Fig. 21 Average ETE delay (with handoff)** **17**

Fig.B-1 Signaling Protocol Structure for IP-based network **24**

Fig.B-2 H.323 terminals on a Packet Network **25**

Fig.B-3 H.323 Call Establishment **Fig.B-4 H.323 Call Release** **26**

Fig.B-5 Handoff procedure in H.323 ..... **27**

1. Abstract

In recent years, Wireless Local Area Networks (Wireless LAN or WLAN) is becoming a hot topic in both research and commercial field. There is no surprise with this trend because of the quick market extension in wireless data communication and efficiency and convenience that people can obtain from the high wireless technology. One of the most important issues is the mobility management within a WLAN subnet. Many alternative ways to realize mobility support have been proposed on different layers. Mobility support using Session Initial Protocol (SIP) is proposed in order to support real-time communication in a more efficient way in application layer.

In our project, we are interested in simulating a handoff procedure based on SIP protocol over Wireless LAN. We will study SIP protocol and OPNET models for Wireless LAN in detail, and also refer some current research works concerned about the topic. By modifying the contributed processes models in OPNET, we try to set up a simulation environment in which we can simulate the handoff process supported by SIP.

Keywords: Mobile Host (MH), Correspondent Host (CH), Basic Service Set (BSS), Extended Service Set (ESS),

2. Introduction

One of the major stumbling blocks in the efforts toward the development of Wireless Internet Infrastructure is the mobility management in such an infrastructure. Several mobility support methods have been proposed. In [2], mobile IP has been proposed as a solution for mobility support in IP networks. A well-known problem with the mobile IP is the triangular routing, which has been deal with in [8] by route optimization. On one hand, route optimization solves the triangular routing problem by using binding updated to inform the correspondent host about the current IP address of mobile host; on the other hand, it brings about several new problems stated in [1]. Protocols have also been proposed to provide upper layer mobility such as MSOCKS in [9] or TCP Migrate in [10], which provides transport layer mobility. All these methods have their own restrictive scope of operation.

Besides all above mobility support methods, analysis show that a number of most popular applications in the Internet, such as email, messaging, etc., can be supported by simple application layer mobility. Session Initial Protocol (SIP) [4] is an application layer text based signaling protocol. It is used to establish and tear down multimedia sessions. The fact that SIP has been already accepted as the signaling standard in 3G wireless systems makes it attractive for providing complete solution to signaling and mobility management in 3G wireless networks and telecommunication systems. In other words, mobility management has been integrated as a part of signaling protocol, without any extra cost or load.

In our project, we aim at building a simple source-to-destination voice session via a proxy server based on SIP protocol in OPNET. And we study the session setup and tearing down procedure. Then, we will enhance the functionality of SIP-related process models in order to support mobility management. Finally, we will analyze the results and evaluate whether the delay introduced by handoff procedure is acceptable in the real-time traffic, such as voice, stream media.

2.1 Wireless LAN and IEEE 802.11 Standard

A wireless local area network (WLAN) gives the functionality of a traditional wired infrastructure to users but with more flexibility. It is a type of local-area network that uses high-frequency radio waves rather than wires to communicate between nodes. WLAN is a flexible data communication system seen as an alternative to, or an extension of a wired LAN. WLAN transmits and receives data over the air using electromagnetic waves, minimizing the need for wired connections. And more important thing is that WLAN users can access information and network resources as they attend meetings or when they move to other campus locations. Thus, WLAN combines data connectivity with user mobility, and, through simplified configuration, enable movable LAN.

The 802.11 standard protocol family defined by IEEE (Institute of Electrical and Electronics Engineers), gives detailed specifications on the parameters of both the physical (PHY) and medium access control (MAC) layers of the network. It also makes provisions for data transmission rates from 1 Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps and up to 54 Mbps operating in the 2.4 GHz band and 5 GHz band respectively in 802.11a and 802.11b. There are two different kinds of architectures to configure a network: ad-hoc and infrastructure. In an Ad Hoc network, workstations have a peer-to-peer connection with other stations but the communication will be limited to this network. In the later architecture, a network access point (AP) is used with which mobile nodes can communicate to each other or to wired nodes. See Fig.1 and Fig.2.

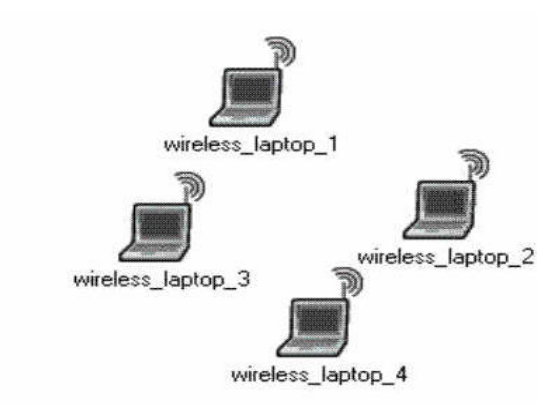


Fig.1 Ad Hoc network

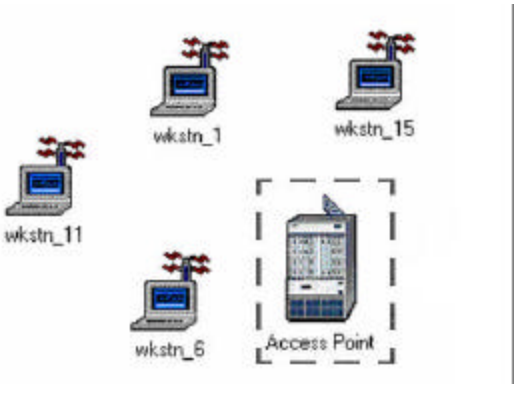


Fig.2 Infrastructure BSS

2.2 Wireless LAN Mobility

Mobile hosts can move about freely and access a WLAN subnet from almost anywhere within the service area of its binding AP. And with the cooperation with other protocols, users can establish their connection to Internet even in different AP service area (Foreign Network) within the same subnet. Being very similar to the current cellular networks around the world, once the cover ranges of AP service overlap, handoffs can occur while user moving from one AP to another.

2.3 SIP overview

The Session Initiation Protocol (SIP), defined by Internet Engineering Task Force (IETF), is an application-layer control (signaling) protocol for establishing, modifying and terminating sessions with one or more participants. These sessions may be Internet telephone calls, multimedia distribution or multicast conferences. It has been standardized within IETF for the invitation to multicast conferences and Internet telephone calls. In the context of SIP, a user is usually identified by an email-like address such as user@domain, where "user" is a user name or phone number, and "domain" is a domain name or numerical address of the user.

The main entities in SIP are User Agent (UA), Proxy Server, Redirect Server and Registrar:
A **User Agent** (UA) is the endpoint entity. User Agents initiate and terminate sessions by exchanging requests and responses. RFC 2543 defines the User Agent as an application, which contains both a user agent client and user agent server as follows:

User Agent Client (UAC)—a client application that initiates SIP requests.

User Agent Server (UAS)—a server application that contacts.

A **Registrar** is a server that accepts REGISTER requests for the purpose of updating a location database with the contact information of the user specified in the request.
A **Proxy Server** is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients.
A **Redirect Server** is a server that accepts a SIP request, maps the SIP address of the called party into zero (if there is no known address) or more new addresses and returns them to the client.

The message types of SIP are typically Request-Response messages, either a request from a client to a server, or a response from a server to a client. Currently, there are six methods of request defined in SIP as follows:

Table.1 SIP Request method

Message Name	Function
REGISTER	Register a user with a SIP server (with location service)
INVITE **reINVITE**	Invite user(s) to a session. The body of the message contains the description with the address where the host wants to receive the media stream ReINVITE is for changing the session (call) parameters
ACK	Acknowledgement of an INVITE request
CANCEL	Cancel a pending request
BYE	Terminate a session (release a call)
OPTIONS	Query servers about their capabilities

Responses to request methods by a three-digit status code indicate success or failure, distinguished by code as follows:

Table. 2 SIP Response code

Code Classes	**Response Type**	**Function Description**
1xx	Provisional	Request received, continuing to process the request
2xx	Success	The action was successfully received, understood and accepted
3xx	Redirection	Further action needs to be taken in order to complete the request
4xx	Client Error	The request contains bad syntax or cannot be fulfill at this server
5xx	Server Error	The server failed to fulfill an apparently valid request
6xx	Global Failure	The request cannot be fulfill at any server

SIP messages are composed of the following three parts: START LINE; HEADERS and BODY (CONTENT). SIP messages appear both in request and in response messages. SIP makes a clear distinction between signaling information, conveyed in the SIP Start Line and headers, and the session description information.

generic-message = start-line (= Request-Line | Status-Line)

message-header = (general-header
| request-header
| response-header
| entity-header)
CRLF (Carriage-Return Line-Feed: an empty line indicating the end of the header fields)
[message-body]

Every SIP message begins with a Start Line. The Start Line conveys the essage type (method type in requests, and response code in responses) and the protocol version. The Start Line may be either a Request-line (requests) or a Status-line (responses), such as follows:

-- The Request-line includes a Request URI, which indicates the user or service to which this request is being addressed.
-- The Status-line holds the numeric Status-code and its associated textual phrase.

SIP header fields are used to convey message attributes and to modify message meaning. Headers can span multiple lines. Some SIP headers such as Via, Contact, Route and Request-Route can appear multiple times in a message or, alternatively, can take multiple comma-separated values in a single header occurrence.

A message Body is used to describe the session to be initiated or alternatively it may be used to contain opaque textual or binary data of any type, which is related in some way to the session. Message bodies can be written as
<name> :< value>

Fig.3 shows the interaction between a User Agent Client (UAC) and a User Agent Server (UAS) establishment and termination during the session. And for each SIP session, there is a unique call identifier (Call-ID) that identifies the session. If the session needs to be modified, the same Call-ID is used in the initial request, in order to indicate that this is a modification of an existing session.

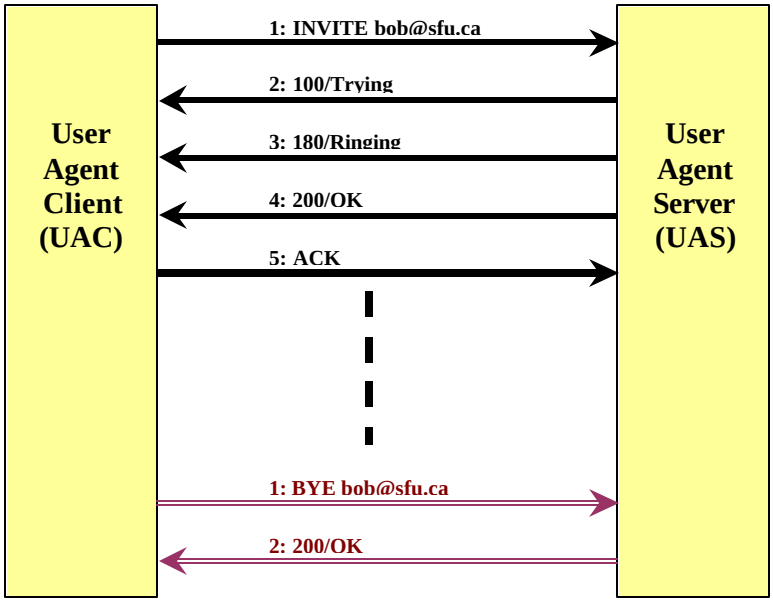


Fig.3 SIP Session Establishment and Call Termination

2.4 Mobility Support using SIP over WLAN

The user in a SIP session may move between different access points' service area over time. These locations can be dynamically registered with the SIP server. According to the dynamic parameter, SIP can initial a new session, or can also modify the existing session without releasing the session. These functions give a support to user's mobility over WLAN for which we will introduce in our project.

A typical handoff procedure in application layer mobility management includes two phases. One is registration in the new location; the other is redirection of an ongoing session, during which considerable delay may be introduced to the session. SIP can support terminal, session, personal and service mobility [11]. Terminal mobility allows a device to move between different Access Points while being reachable to other hosts and continuing any ongoing session while on the move. It requires set-up of a connection either during the start of a new session when the user has already moved to a different location, or in the middle of an existing session. We call the former situation as Pre-Call mobility, and the latter one as Mid-Call mobility. For Pre-Call mobility, the user (Mobile Host: MH) will register, or re-

register its new location with its home registrar SIP server or redirect server. Any call to the MH is thereby forwarded to the new location. By consulting the home server of MH and obtaining the new IP address of MH, the Corresponding Host (CH) establishes the session with MH *directly*. It solves the issue of the triangular routing happened in Mobile IP support Mobility. See Fig.4 for this registration procedure

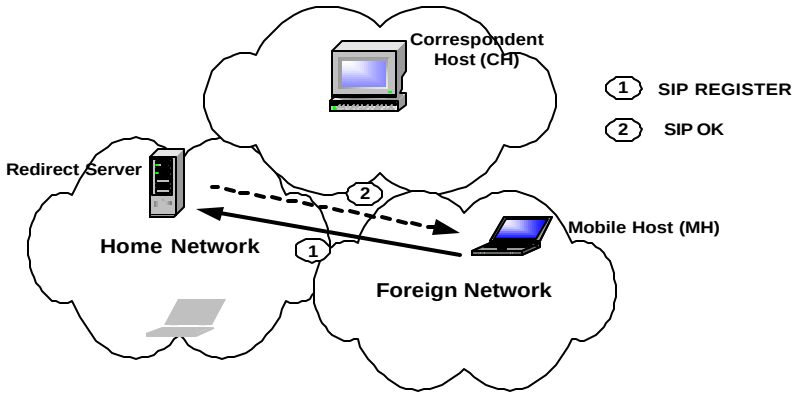


Fig. 4 Pre-Call mobility: registration procedure

In the case of Mid-Call mobility support, the MH will notify the CH about its location change by sending SIP request directly. We already know that SIP client (UAC) can initiate a request to modify an existing session by sending a new INVITE message using the same Call-ID. The MH attaches the update description of the new IP address with the new INVITE (reINVITE) message directly to CH to tell the CH where it wants to receive future message. After MH receives acknowledgement from CH for this reINVITE, the two sides of this connection will continue following data communication, which is shown in Fig. 5. In our project, we will concentrate on the implementation of Mid-Call mobility support.

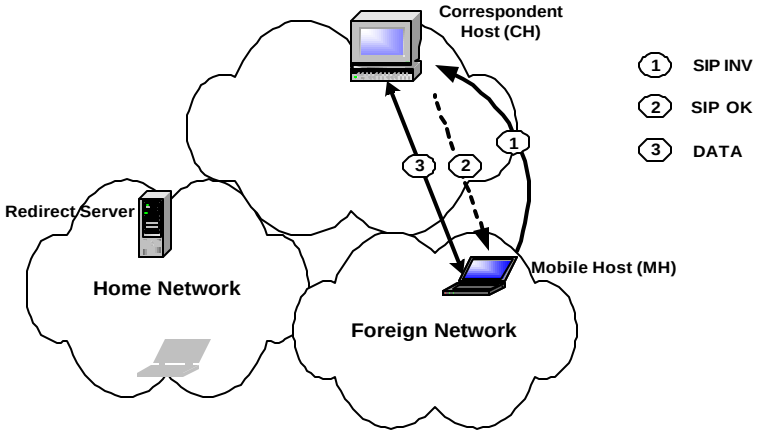


Fig. 5 Mid-Call mobility: reINVITE procedure

According to [3], Fig.6 gives out a detailed description of timing for SIP-based handoff procedure in WLAN.

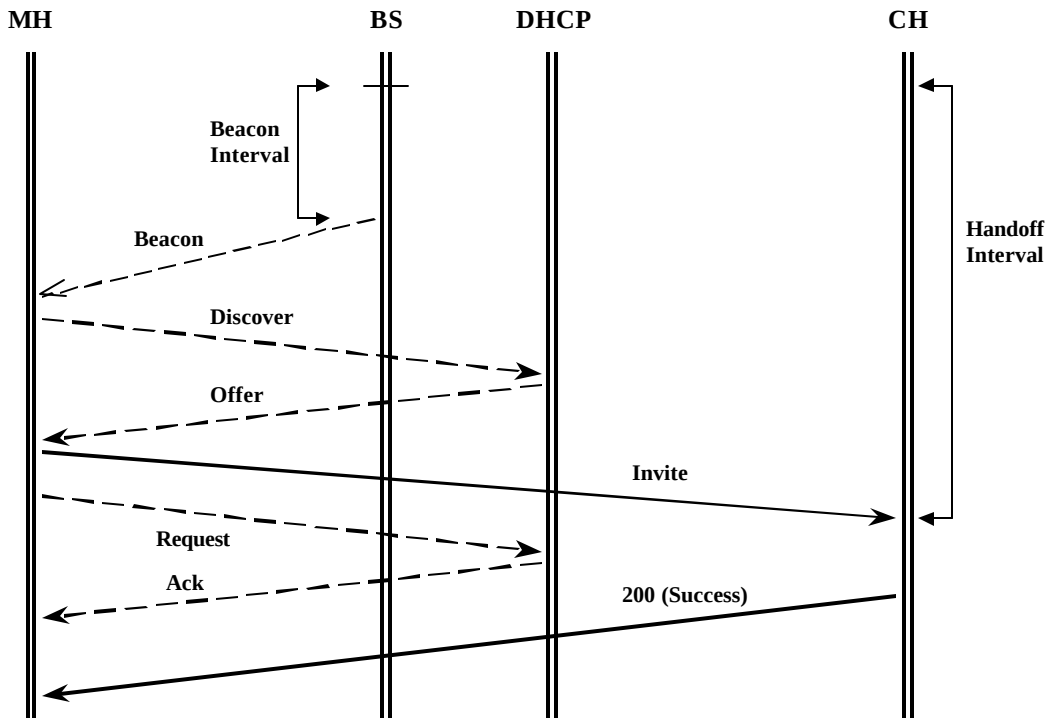


Fig. 6 Typical timing for SIP-based handoff procedure in WLAN

3. OPNET Models for Wireless LAN

3.1 Node models

In OPNET, Wireless LAN node models are given as part of standard OPNET model release and are required for any simulation concerned with WLAN. This model is based on IEEE STD 802.11, 1999 and IEEE STD 802.11b, 1999. This section will describe some main WLAN node structures and network configurations done in OPNET concerned with our project.

WLAN workstation node model, shown in Fig. 7, consists of MAC and physical layer modules, ARP (Address Resolution Protocol) and some higher layer modules, such as TCP/IP and applications. It represents a workstation with client-server applications running over the underlying WLAN connection at the data rate of 1, 2, 5.5 or 11 Mbps.

The WLAN Router, shown in Fig. 8, contains one wireless interface and one Ethernet interface. When it is defined as an Access Point (AP), it can connect a BSS with a wireline distribution system. When the wireless interface receives a data frame that does not belong to the same BSS, it sends the data frame to the higher layer for address resolution.

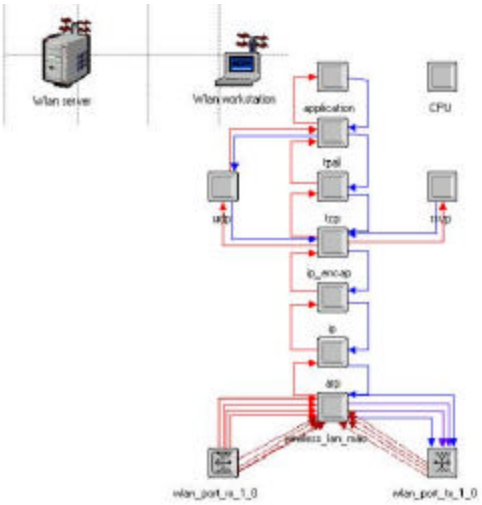


Fig. 7 WLAN workstation

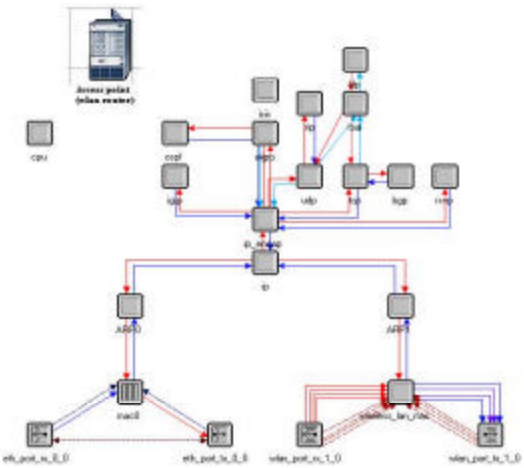


Fig. 8 WLAN Access Point

In our project, we concern more about the ESS, in which workstations communicate with each other and with nodes outside their LAN through Access Point. Here, a WLAN Router is configured as an AP, as shown in Fig. 9 and Fig. 10, acting as interfaces between WLAN and Internet.

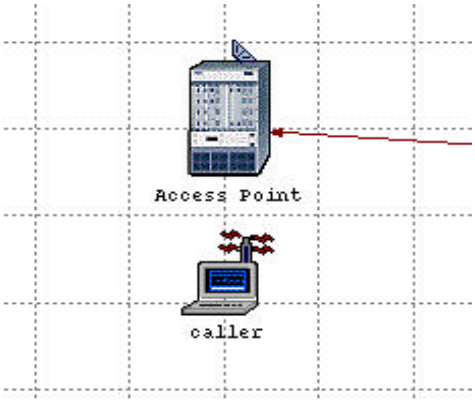


Fig. 9 Caller side subnet

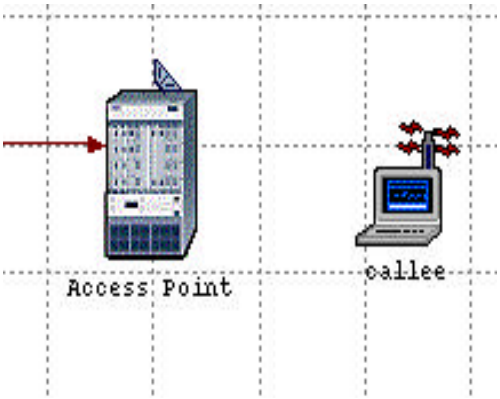


Fig. 10 Callee side subnet

3.2 Interested Statistics

We are interested in some statistics on WLAN performance. Several statistics can be collected during simulation execution. Statistics can be collected on a per-node or a per-module basis. The node statistics we are after:

1) Delay: End-to-End delay of all packets received by the node's wireless LAN MAC and forwarded to the higher layer.

2) Load: Total number of bits received from the higher layer. Packets arriving from the higher layer are stored in the higher layer queue.

4. Project Implementation in OPNET

4.1 Assumptions

In order to concentrate on the handoff procedure itself, we would try to do some simplifications. First, in realistic environment, mobile host usually needs to sense the beacon sent by Access Point. Once it discovers a new DHCP server and gets a new IP address from it. It will send the reINVITE request to the correspondent host. This procedure means we must design a mechanism to detect the variation of MAC layer parameters and arrange an appropriate interrupt according to this event. This is not close related to our work, since we want to simulate the SIP function after this interrupt occurs. So, how it occurs is not so important to us and we will choose to generate this interrupt within application layer using a special timer.

Realistic SIP-based handoff procedure contains a three-way handshake between client and server, as shown in Fig. 6. According to [3], the major portion of delay introduced by SIP-based handoff is due to the time taken to transmit the reINVITE message. So, we will focus on implementing this procedure. Specifically speaking, we will design a special packet for handoff. Once a handoff interrupt occurs, client will send this packet to server. Upon server's receiving this packet and client's receiving of acknowledgement from the server, handoff procedure can be regarded as successfully completed. Future voice data can continue according to the updated binding of client's location information.

4.2 SIP models in OPNET

OPNET has contributed necessary WLAN node models, which have been described above. Besides all this, we need to use SIP related process models and some process models concerned with voice application that uses SIP as its signaling protocol. A list of these process models is given below and we will give a brief introduction to them.

gna_clsvr_mgr: process model for the application module of the WLAN workstation. It is used to manage the generic network application defined in Application Configure, initiating corresponding network application on this workstation node.

gna_profile_mgr: process model that is created by *gna_clsvr_mgr* to manage a profile for each client.

gna_voice_calling_mgr: process model that is specifically related to voice application. It is created by *gna_profile_mgr* to deal with the voice application generated by *gna_clsvr_mgr*.

sip_UAC_mgr: process model that is called by *gna_voice_calling_mgr* when it wants to setup a voice session. A *sip_UAC_mgr* usually holds control over several *sip_UACs* and call status that is mapped to a specific *sip_UAC*.

sip_UAC: process models that are created by *sip_UAC_mgr* to setup a session between a client and a server.

We can clearly see the hierarchy relationship in the order of our introduction. Note that all these process models are located within application layer, from the top to the bottom, which is shown in Fig. 11.

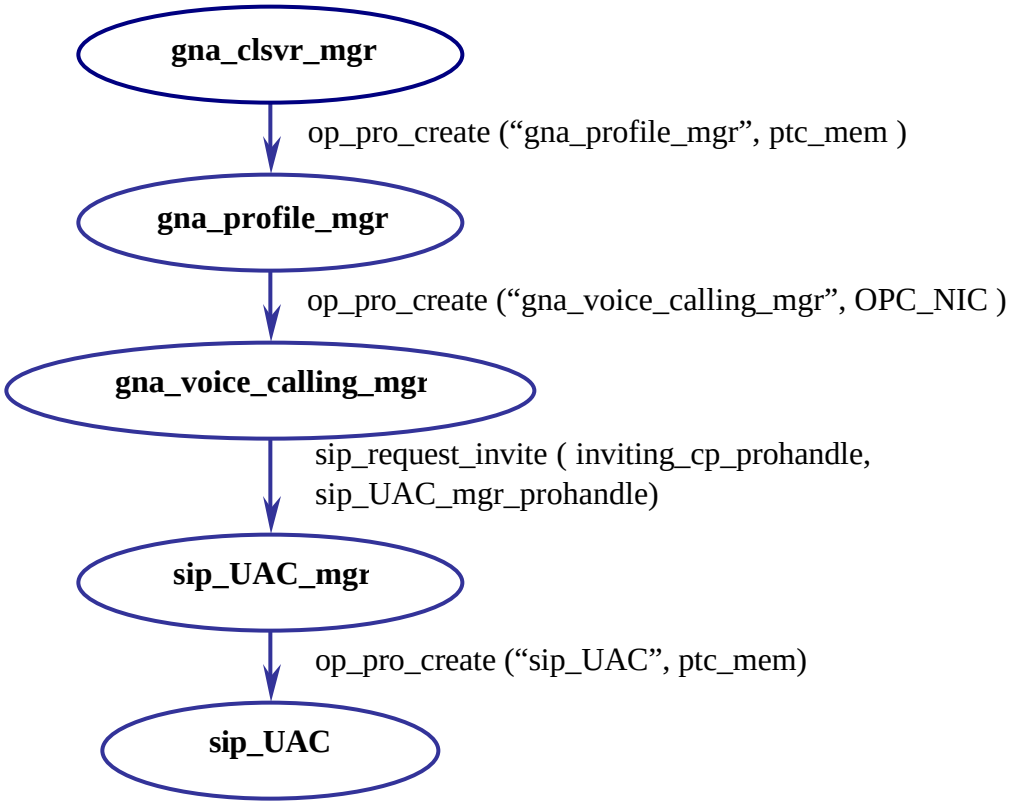


Fig. 11 Hierarchy relationship within application layer

From Fig. 11, we may note that *gna_voice_calling_mgr* is the process model that starts to directly interact with SIP process model. In fact, it keeps a state variable which points to a specific *sip_UAC_mgr*. When this voice-calling manager wants to setup a session with the remote host, it will call the function *sip_request_invite( )*, trying to open and maintain an active connection with the remote side. Using all these models, we built up a network topology to implement the basic SIP-based voice call setup and terminating procedure, which is shown in Fig. 12.

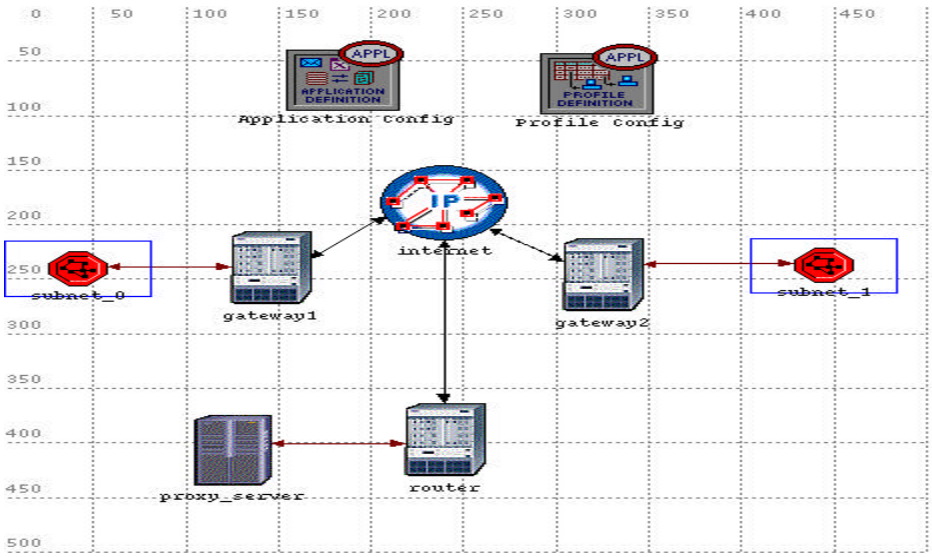


Fig. 12 Network topology

Where subnet_0 and subnet_1 have topology described in Fig. 9 and Fig. 10 respectively.

4.3 Our modifications to SIP process models

gna_clsvr_mgr (generate voice application)

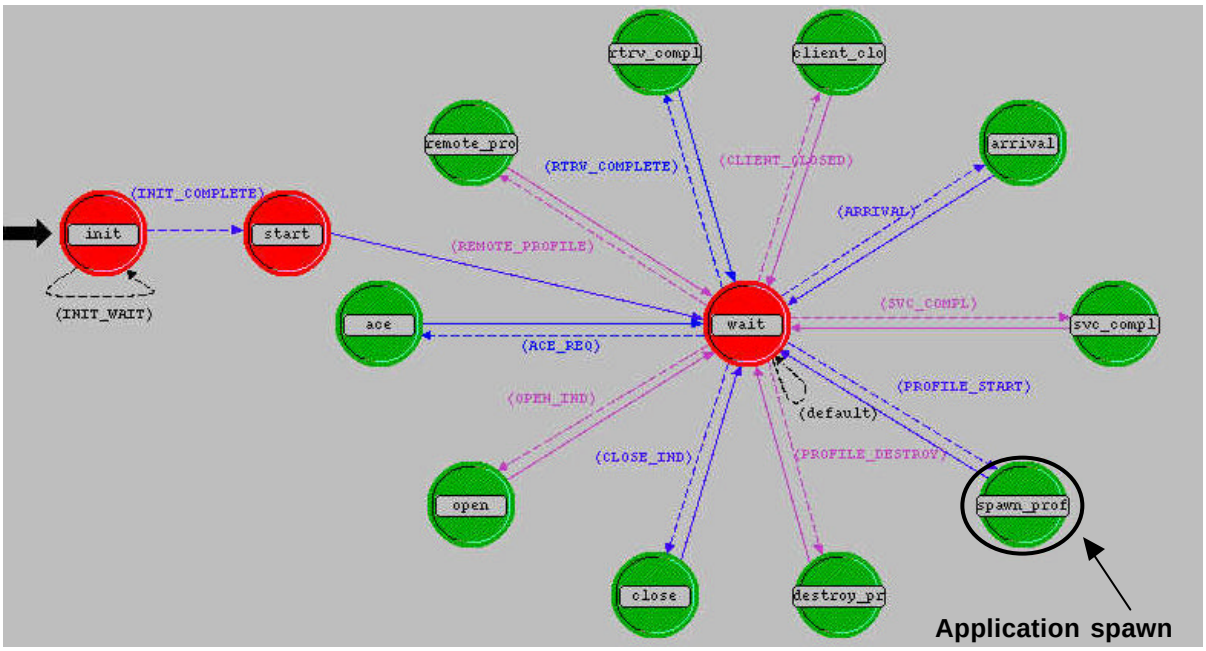


Fig. 13 Process model: gna_clsvr_mgr

gna_profile_mgr (generate a profile for a client)

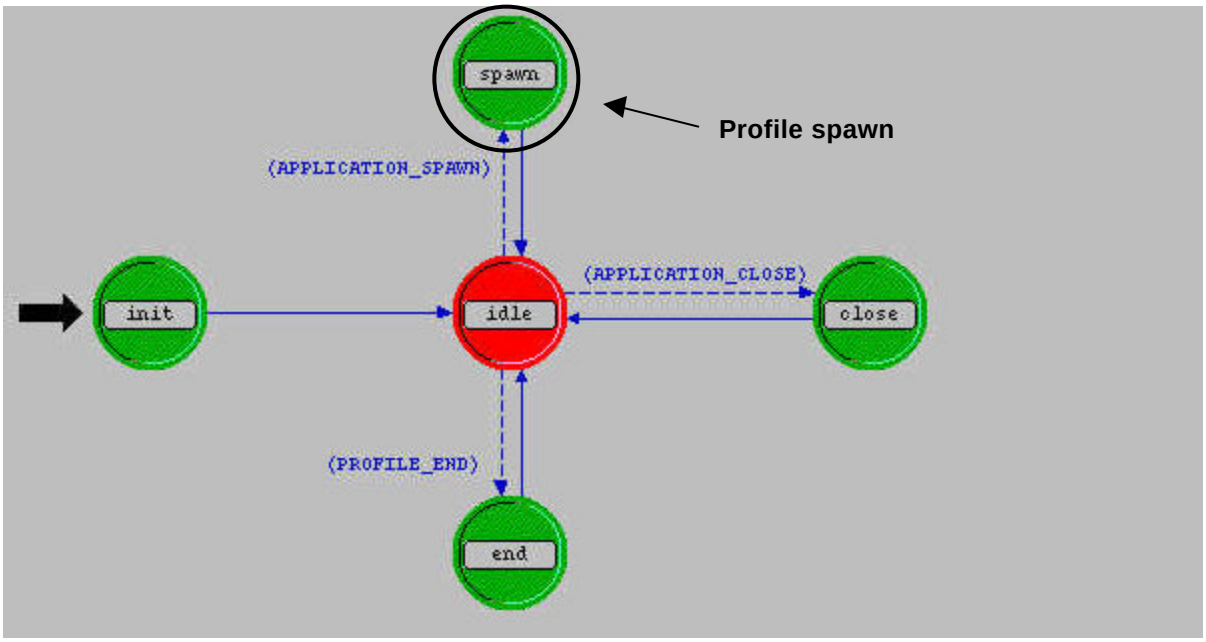


Fig. 14 Process model: gna_profile_mgr

gna_voice_calling_mgr (generate handoff interrupt)

Without handoff occurrance, gna_voice_calling_mgr continues to transfer data once it enters the state “idle”. We try to artificially simulate that a handoff interrupt occurs during the process of sending data. With this event, it will enter the state “handoff” and arrange an interrupt to the sip_UAC related to the same call. The transfer of this interrupt should be realized via sip_UAC_mgr.

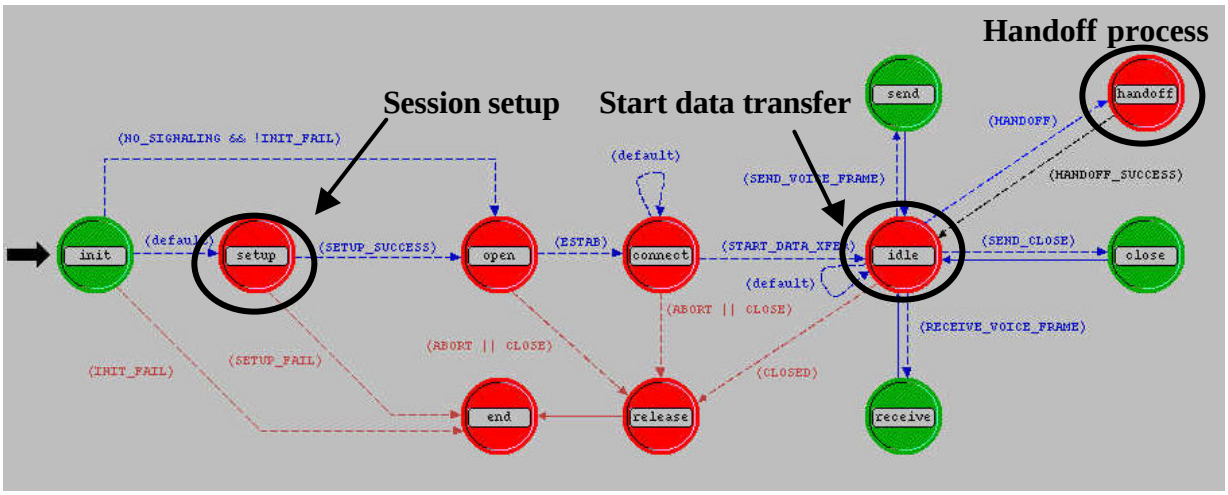


Fig. 15 Modified process model: gna_voice_calling_mgr

sip_UAC_mgr (redirect handoff interrupt to sip_UAC)

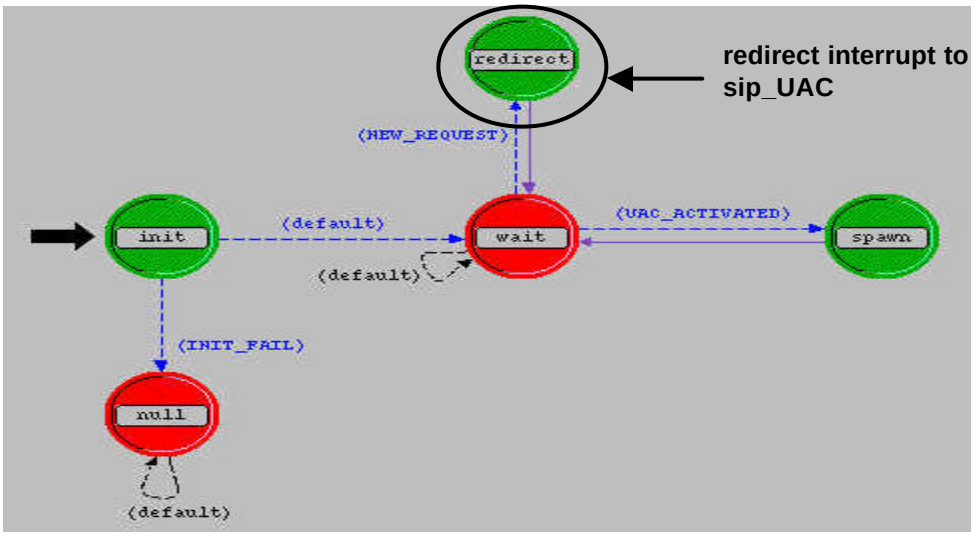


Fig. 16 Process model: sip_UAC_mgr

sip_UAC (receiving the handoff interrupt and react to it)

The modified sip_UAC process model is shown in Fig. 17. After initiation, the client will enter the passive “listen” state to wait for the coming request. Normally, it will process both request and response received from upper layer and remote host respectively. When

receiving a REQ or RESP, the client will be pushed into the “fork” state. In this state, it will choose to process a pending request or response. Usually, response has a priority over request, since response often indicates an active connection waits to be open. After processing all requests and response, the client will enter the end state. What we added is the state “Handoff”. Once sip_UAC receives a handoff interrupt redirected from sip_UAC_mgr, it will leave “listen” state and enter “handoff” state after it sends out a special “handoff” packet to the server side, then begins to wait for the response from the server.

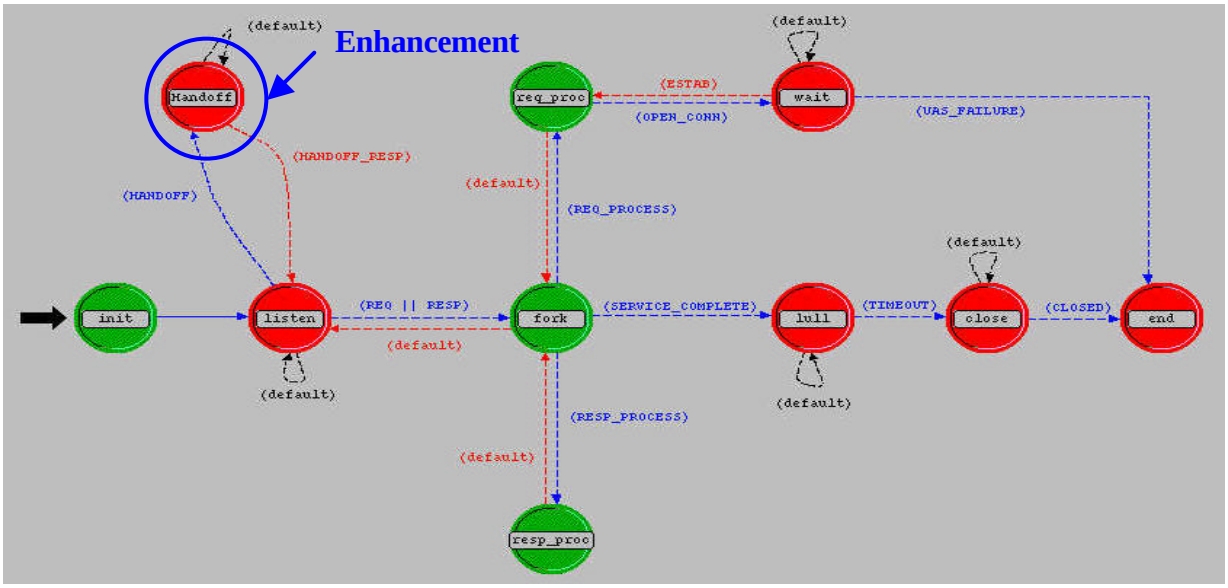


Fig. 17 Modified process model of sip_UAC

sip_UAS (receiving the handoff packet and send back response)

The “init” state opens a passive transport connection. Then UAS will enter “open” state and listen to the incoming requests at port 506. If there is a remote interrupt to setup a session, the UAS will establish an active connection and at the same time inform the UAS manager to spawn a new UAS to listen the requests. After the connection is established, the process then sits in the “listen” state to wait for the further stream packets from the UAC side. If the packet is a sip packet, the UAS will get the message type first, and then take actions according to the type of the message. If the packet type shows an INVITE request, the UAS will call the function block sip_UAS_invite_req_process to process the request. When the incoming request is destined to other destination UAC, the UAS will relay the packet to its destination. According to our project goal mentioned before, we need to assume that when the caller moves to a new Access Point service area, a special type of packet will be sent from the UAC side to the UAS side. When the UAS receives this packet, it will send back a response to UAC indicating the receipt of the packet. After that, the handoff process is regarded as finished. So in the UAS process model, we add a new state named hand_off to simulate and develop the according code to implement this function. Fig. 18 shows the modified UAS process model:

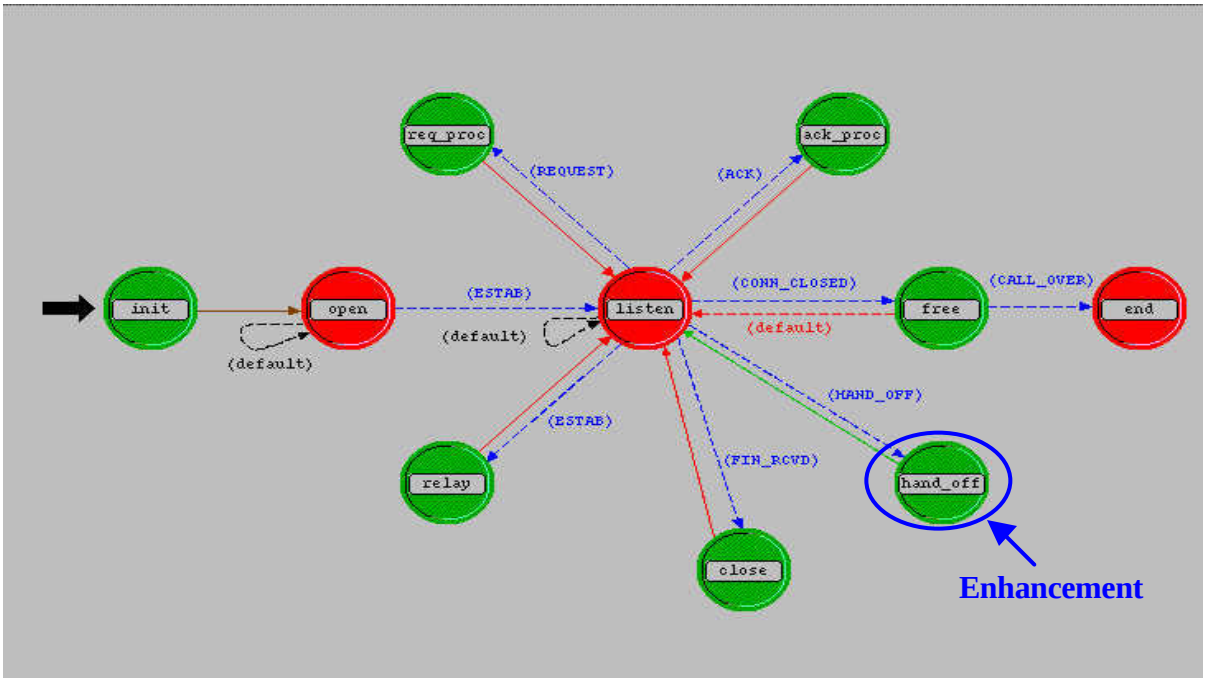


Fig. 18 Process model: sip_UAS

5. Simulation Results and Problems still existing

This section describes the simulations that have been run using the network topology presented above. Voice application parameters are as follows. Note that we set SIP as the signaling protocol.

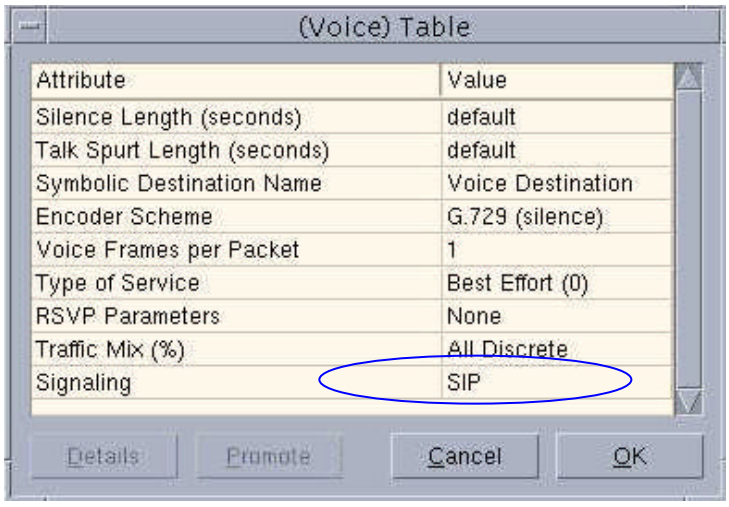
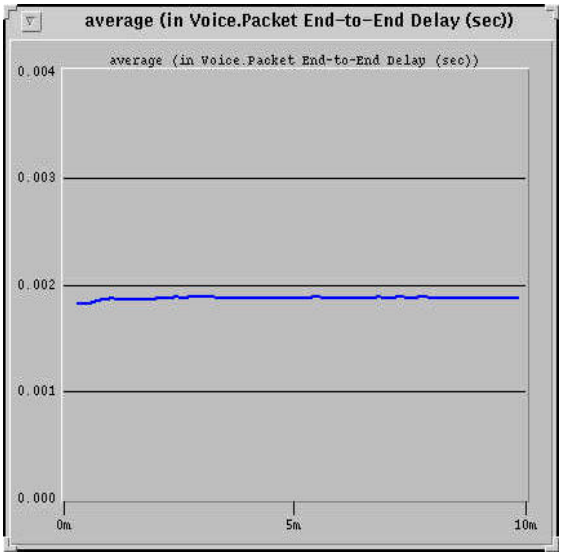


Fig. 19 Voice application configuration

The following is the average End To End delay of voice packet in static (without handoff) and the situation of handoff occurrence.



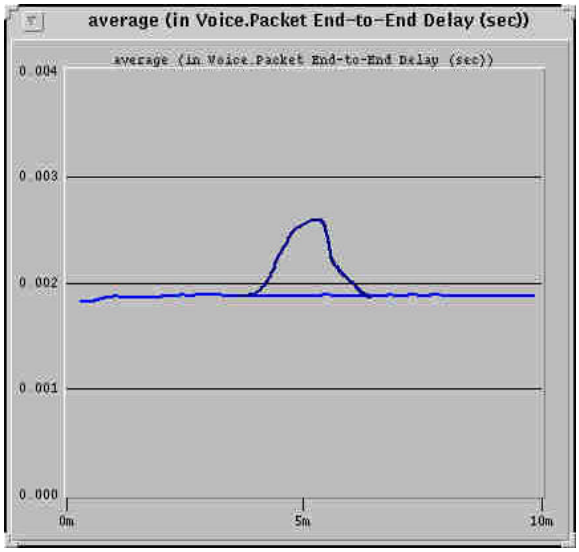


Fig. 20 Average ETE delay (no handoff) **Fig. 21 Average ETE delay (with handoff)**

Problems still existing:

Our original idea is to use "trajectory" defined in OPNET to simulation a realistic curve for the movement of the caller. Simultaneously, in MAC layer, we detect the variation of receiving signal power and schedule an interrupt when the power is below a pre-defined threshold. However, as we go further, we find SIP functionalities all locating within application layer. Hence it makes no difference whether this interrupt is triggered from the low layer or upper layer, so we concentrate on how to react to this interrupt, not how to generate this interrupt.

As mentioned, for simplification, we choose to artificially set a timer in *gna_voice_calling_mgr* to schedule this interrupt. Then we need to transfer this interrupt to sip_UAC via sip_UAC_mgr. The current major difficulty for us is how to manage this interrupt and react to it throughout all the above hierarchy process models. We start our coding work from the bottom layer, i.e. sip_UAC and sip_UAS. The programming work in these two models has been finished and codes will be provided in Appendix A. However, we still owe some debugging work in upper layer process models within application layers, i.e. gna_voice_calling_mgr.

6. Discussion and conclusion

In this project, we try to implement a scenario to simulate SIP-based handoff procedure over wireless LAN. Most of our project has been completed, although some further work needs to be done to handle "Ici" in OPNET, which is related to interrupt processing in the upper part within the application layer.

Theoretical analysis in [3] shows that SIP-based handoff procedure has an approximate delay of 6 seconds. However media streams can function normally with a maximum interruption of 50 msecs, which means SIP-based handoff procedure over wireless link can not satisfy the delay requirement of stream media traffic. Simulation results need to be further carried out to

verify this conclusion. Due to the lack of time, we feel regret that we can not reach a final simulation result validate our analysis. But by doing this project, we have learned a lot on the software architecture of OPNET and how to build our own simulation scenario. We also learned a lot on how to find the process models that we need to add our own codes and how to write them. Thanks to the instructor Ljiljana Trajkovic and our teaching assistant Kenny and Wan.

7. Future work

In this project, we only concentrate on the part of mid-call mobility support in our project. Maybe, future work can be done on how to implement the pre-call mobility management, i.e. registration procedure. Also, a possible future development of this project will be to simulate realistic handoff scenario as what we have originally considered using “trajectory”.

Reference:

[1] E. Wedlund, H. Schulzrinne, "Mobility Support using SIP", ACM/IEEE International Conference on Wireless and Multimedia (WOWMOM), Aug. 1999, pp. 76-82.

[2] James F. Kurose, Keith W. Ross, "Computer networking: a top-down approach featuring the Internet", Addison-Wesley, 2nd edition, 2003

[3] Banerjee N., Basu K., Das S.K., "Hand-off delay analysis in sip-based mobility management in wireless networks", Parallel and Distributed Processing Symposium, 2003. Proceedings. International , April 22-26, 2003.

[4] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, Internet Engineering Task Force, March 1999., SIP: Session Initial Protocol

[5] IEEE, "802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", approved 26 June 1997.

[6] OPNET documentation: "Wireless LAN Model Description", MD-36-1 to MD-36-16, Release 9.1

[7] Abdul Mukti Bin and Mohd Dani Bin Baba, "Handover Strategy for Mobile Wireless LAN", 4th National Conference on Telecommunication Technology Proceedings, 2003

[8] H. Chen, Ljiljana Trajkovic, "Route optimization in mobile IP", Proc. Workshop on Wireless Local Networks (WLN) 2002, Tampa, FL, Nov. 2002, pp. 847-848.

[9] D. Maltz and P. Bhagwat, "MSOCKS: an architechture for transport layer mobility", Proceedings of INFOCOM, Page(s): 100-108, April 1998

[10] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility", Proceedings MobiCom, Pages: 155-166, August 2000.

[11] H. Schulzrinne and E. Wedlund, "Application-Layer Mobility Using SIP", Mobile Computing and Communication Review, Vol.1, No.2, Pages: 47-57.

Appendix A: code listing

1 Modified header file
sip_support.h
sip_api.h

2 Modified process models
1) sip_UAC added function

```
/* this function will send a handoff packet to UAS */
static void
sip_UAC_send_handoff_to_UAS (Ici* sip_req_ici_ptr)
        {
        char                                    msg [128];
        Packet*                                 sip_pk_ptr;
        SIPT_Call_Info_Shell*                   sip_call_info_shell_ptr;
        SIPT_Call_Info*                         sip_call_info_ptr;

        /* This function processes the HANDOFF request */
        FIN (sip_UAC_send_req_to_UAS (sip_req_ici_ptr));

        /* Get the process handle of the caller from thc ICI */
        op_ici_attr_get (sip_req_ici_ptr, "appl_prohandle_ptr", &appl_prohndl_ptr);

        /* Get the call information from thc ICI */
        op_ici_attr_get (sip_req_ici_ptr, "call_info_ptr", &sip_call_info_shell_ptr);
        sip_call_info_ptr = sip_call_info_shell_ptr->sip_call_info_ptr;

        /* Save the call handle (shell) for reference */
        call_info_shell_ptr = sip_call_info_shell_ptr;

        /* Map the call to this UAC so thatthe UAC_mgr */
        /* knows which call is handled by which UAC    */
        sip_map_call_to_UAC ();

        /* Create a SIP packet to communicate the request to the UAS */
        sip_pk_ptr = op_pk_create_fmt ("sip");

        /** Update the fields of the packet **/

        /* Set the type of the packet to indicate a HANDOFF request */
        op_pk_nfd_set (sip_pk_ptr, "type", SIPC_Packet_Type_Request);

        /* Set the message to indicate a Call Handoff */
        op_pk_nfd_set (sip_pk_ptr, "msg", SIPC_CALL_HANDOFF);

        /* Set the call_id */
        op_pk_nfd_set(sip_pk_ptr,"call_info",sip_call_info_shell_ptr,sip_call_info_fdstruct_
copy_proc, sip_call_info_fdstruct_destroy_proc, 0);
```

```
	if(ltrace_sip_UAC_active||ltrace_sip_UAC_invite_active||ltrace_sip_UAC_req_active
)
		{
		sprintf(msg, "Sending the request to UAS in packet (PK_ID %f)", op_pk_id
(sip_pk_ptr));
		op_prg_odb_print_major(msg, OPC_NIL);
		}

	/* Install the session ici */
	op_ici_install (session_ici_ptr);

	/* Send the packet to the UAS */
	op_pk_send (sip_pk_ptr, strm_to_tpal);

	/* Destroy the request ICI */
	op_ici_destroy (sip_req_ici_ptr);

	/* Reset he counter for expected responses */
	expected_resp_count++;

	FOUT;
	}/* End sip_UAC_send_req_to_UAS ()*/
```

2) gna_voice_calling_mgr added function

```
static void
gna_voice_sip_call_handoff ()
	{
	char				msg [256];
	GnaT_Voice_Desc*		voice_info_ptr;

	/* This function sets up a call using SIP */
	FIN (gna_voice_sip_call_handoff ());

	/* Get the SIP UAC */
	sip_UAC_mgr_prohandle_ptr = mod_mem_ptr->sip_UAC_mgr_prohndl_ptr;

	if (sip_UAC_mgr_prohandle_ptr == OPC_NIL)
		{
		/* Print an error message */
		gna_voice_sip_service_unavailable_log_write(application_info_ptr-
>application_comp_ptr->application_ptr->application_name_ptr);

		/* Schedule a self interrupt to transit to the END state */
		op_intrpt_schedule_self (op_sim_time (), SIPC_CALL_CONNECT_FAIL);
    }

	else
```

```
		{
		/* Send an HANDOFF request to the SIP UAC */
		if (ltrace_signaling_active)
			{
			sprintf(msg, "Voice Client (PID %d): Sending HANDOFF Request to
SIP UAC", op_pro_id (op_pro_self ()));
			op_prg_odb_print_major (msg, OPC_NIL);
			}

		/* Send a Handoff request */
		sip_handoff_success = sip_request_handoff (op_pro_self(),
*sip_UAC_mgr_prohandle_ptr, remote_host);

		/* Check if the handoff is successful finished */
		if (sip_handoff_success == OPC_FALSE)
			{
			/* handoff failed - setup a self interrupt to transit to the END state */
			op_intrpt_schedule_self (op_sim_time (), SIPC_HANDOFF_FAIL);
			}
		}
	FOUT;

	} /* End gna_voice_sip_call_handoff () */
```

3) sip_UAS added function

```
static void
sip_UAS_handoff_req_process (Packet* pk_ptr)
	{
	SIPT_Call_Info_Shell* sip_call_info_shell_ptr= OPC_NIL;
	SIPT_Call_Info*	sip_call_info_ptr= OPC_NIL;
	Ici*		session_ici_ptr;

	/* this function processes the handoff request.*/
	FIN (sip_UAS_handoff_req_process());

	/* set the packet msg_index field to indicate HANDOFF */
	op_pk_nfd_set (pk_ptr, "msg", SIPC_CALL_HANDOFF);

	/* set the packet type field to RESPONSE */
	op_pk_nfd_set (pk_ptr,"type",SIPC_Packet_Type_Response);

	/* get the call info from the Ici */
	 op_pk_nfd_access (pk_ptr, "call_info",&sip_call_info_shell_ptr);
	sip_call_info_ptr= sip_call_info_shell_ptr-> sip_call_info_ptr;

	/* get the transport session to the src UAC */
```

```
	session_ici_ptr=sip_UAS_session_ici_from_addr_get (sip_call_info_ptr-
>inviter_addr);
	op_ici_install (session_ici_ptr);
	op_pk_send (pk_ptr,strm_to_tpal);

	FOUT

		} /* End sip_UAS_handoff_req_process () */
```

Appendix B: H.323 protocols suite

Our original project scope covers H.323 protocols suite. We originally plan to implement handoff procedure based on both SIP and H.323, and compare their performance. (Handoff based on H.323 will be introduced in this appendix too.) As we explored into more materials, we can not found the implementation of H.323 in current OPNET. Since H.323 includes a bunch of protocols, that is why we call it a protocol suite, we decide to narrow down the topic of our project to SIP. Here is a description of what we have done on H.323.

Same as SIP protocol, H.323 standard family is also an application layer signaling protocol providing a foundation for audio, video, and data communications across IP-based networks, including the Internet, which is shown in Fig.B-1[4].

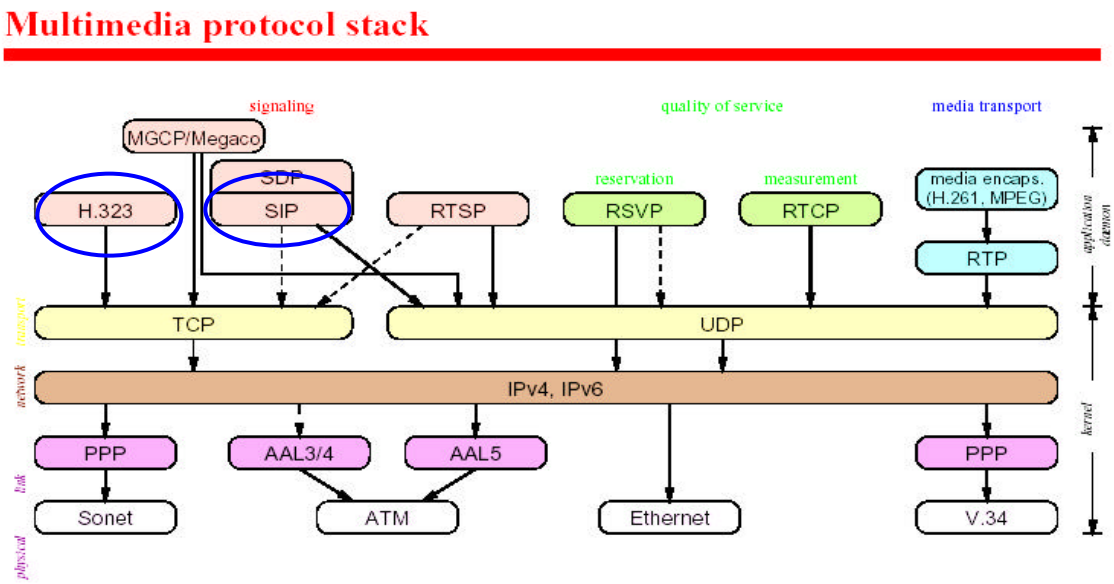


Fig.B-1 Signaling Protocol Structure for IP-based network

Introduction of H.323

Defined by International Telecommunication Union (ITU) - Telecommunication Standardization Sector (ITU-T), The H.323 standard [2] is a cornerstone technology for the transmission of real-time audio, video, and data communications over packet-based networks. It specifies the components, protocols, and procedures providing multimedia communication over packet-based networks. Packet-based networks include IP–based (including the Internet) or Internet packet exchange (IPX)–based local-area networks (LANs), enterprise networks (ENs), metropolitan-area networks (MANs), and wide-area networks (WANs).

H.323 can be applied in a variety of mechanisms—audio only (IP telephony); audio and video (video telephony); audio and data; and audio, video and data. H.323 can also be applied to multipoint-multimedia communications. H.323 provides myriad services and,

therefore, can be applied in a wide variety of areas—consumer, business, and entertainment applications.

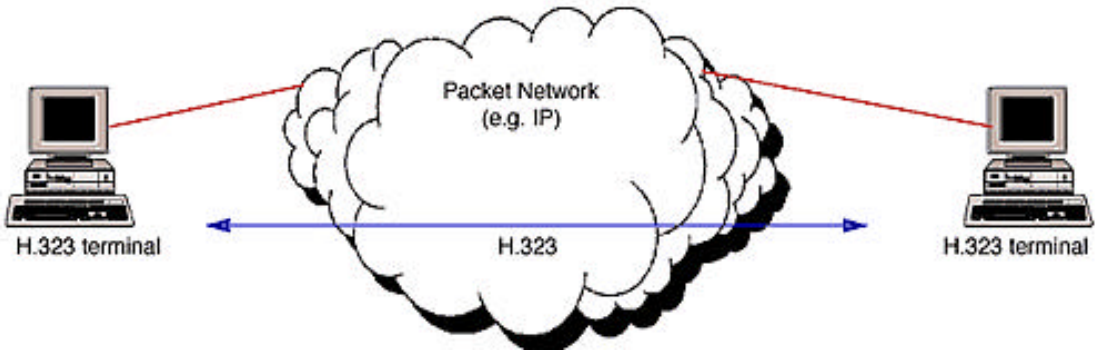


Fig.B-2 H.323 terminals on a Packet Network

It consists of a number of protocols such as H.225, H.245, RTP/RTCP, and some other CODECs which we will introduce later.

H.323 Components

The H.323 standard specifies four kinds of components which provide the point-to-point or point-to-multipoint multimedia-communication services. These four components are:

1. Terminal: Used for real-time bidirectional multimedia communications. A H.323 terminal can either be a personal computer (PC) or a stand-alone device, running a H.323 and multimedia applications.
2. Gateway: Used to connect two dissimilar networks. A H.323 gateway provides connectivity between a H.323 network and a non-H.323 network. It is not required for communication between two terminals on H.323.
3. Gatekeeper: Considered as the brain of the H.323 network. It is the focal point for all calls within the H.323 network.
4. Multipoint Control Units (MCUs): Used for providing support for conferences of three or more H.323 terminals.

An H.323 zone is a collection of all terminals, gateways, and MCUs managed by a single gatekeeper.

Protocols Specified by H.323

The following protocols are part of protocols specified by H.323 [5]. H.323 family protocol is independent of the packet network and the transport protocols over which it runs.

- Audio CODECs
- Video CODECs
- H.325 registration, admission, and status (RAS)
- H.245 control signaling
- Real-time transfer protocol (RTP)
- Real-time control protocol (RTCP)

Connection Procedures in H.323

A typical call set-up is a three-step process which involves call signaling, establishing communication channel for signaling and establishing media channels. The following two figures describe the typical procedures and protocol commands involved in creating call signaling and releasing a H.323 call between two terminals.

The figures only show the signaling relationship between two terminals, but not concern about other components and other protocol commands in H.323 zone [5].

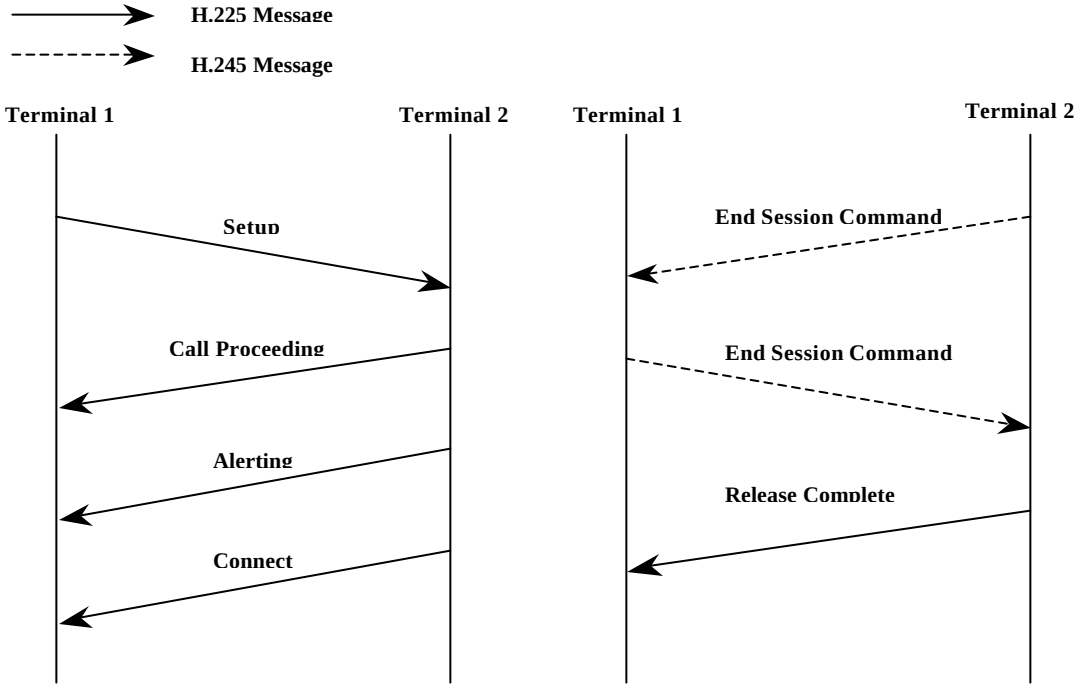


Fig.B-3 H.323 Call Establishment **Fig.B-4 H.323 Call Release**

Handoff Process based on H.323 over WLAN

As mobility support becomes more and more important in current Internet communications, ITU has been developing and standardizing an extension to H.323 for mobility support, which is H.323 annex H [1].

In H.323 mobility architecture and protocol defined by ITU, Multipoint Control Unit will work to set up a new connection in terminal's new location. As a mobile terminal moves from one cell (H.323 zone) to another, a home gatekeeper should keep the current location of its terminal and keep the old connection. When the terminal enters another cell during a call, a new H.323 connection will be established using MCU. And after new call setup, the old one is terminated [3]. Figure 5 shows the process of H.323 handoff procedure [1].

Fig. B-5 describes a typical handoff process defined by H.323 annex H and also gives some main protocols and commands which are involved in this process. There is an existing session (a call) between two terminals (Terminal 1 and Terminal 2). When Terminal 1 moves

from the cell of a Gatekeeper (Gatekeeper 1) to another cell during the session, a handoff process occurs.

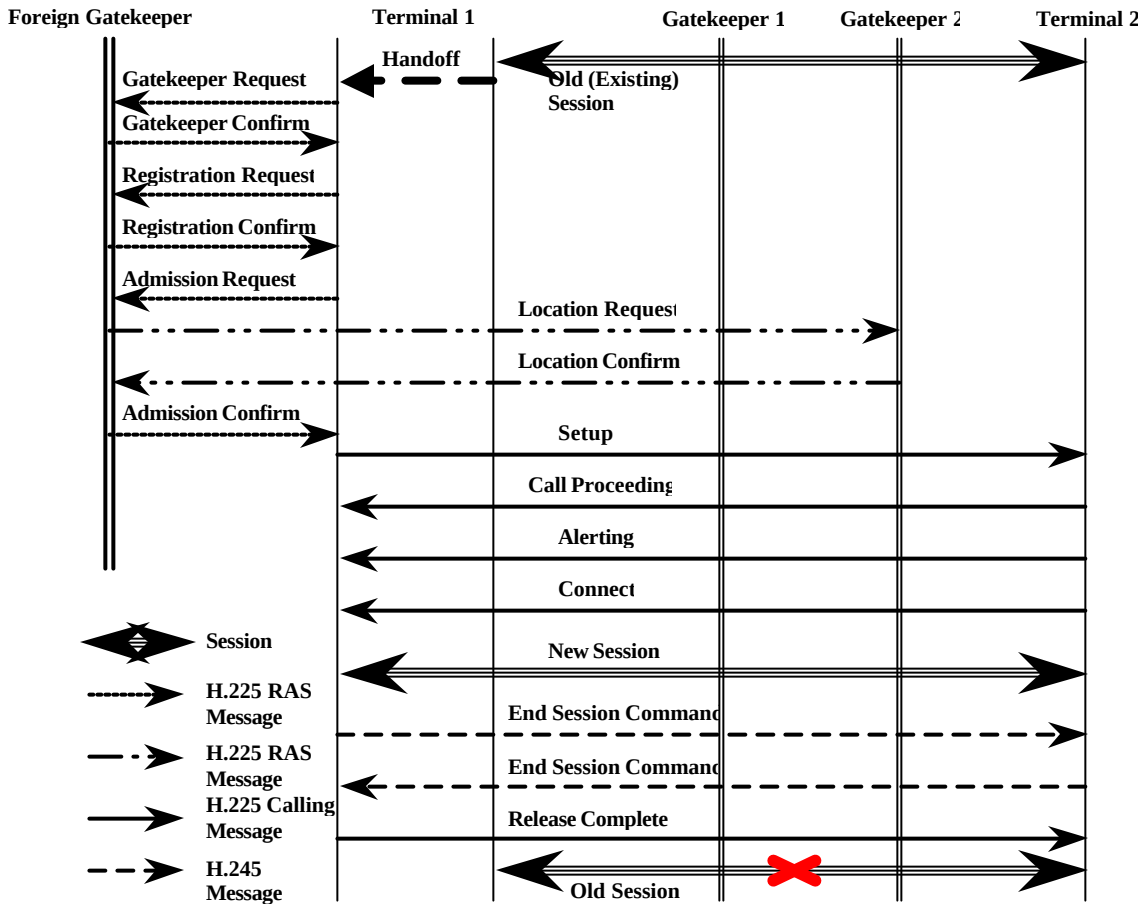


Fig.B-5 Handoff procedure in H.323

Comparison between H.323 and SIP

We can find that the handoff handled by H.323 depends on Multipoint Control (MC) functionality. The terminal can not handle the handoff if the correspondent terminal has not MC function. Since a new connection established during the handoff, it maybe causes the session disruption if the new connection is not created completely, but the old session already is terminated while the moving terminal enters into the new cell. Furthermore, since there are two H.323 connections existed during the handoff process, it occupies the more bandwidth. The handoff process supported by SIP dose not need to set up another session during handoff occurs in an existing session. It just needs to modify the existing session parameters to make the session kept while the user moves from one Access point to another.

References for H.323

[1] DongSeon Park, Wonyong Yoon, Dongman Lee, “An Efficient Handoff Management for Mobility Support to H.323”, International Workshop on Mobile Multimedia Communications, Oct. 2000.

[2] ITU-T Recommendation H.323v2 “Packet-based Multimedia Communications Systems”, Feb. 1998.

[3] Wanjiun Liao, “Mobile Internet telephony: Mobile extension to H.323,” IEEE InfoCom. 1999.

[4] Henning Schulzrinne, “The Session Initiation Protocol (SIP)”, Dept. of Computer Science, Columbia University.

[5] ITC-Web-ProForum, “H.323-Web ProForum Tutorials”, The International Engineering Consortium.