ENSC 835-3: NETWORK PROTOCOLS AND PERFORMANCE
CMPT 885-3: SPECIAL TOPICS: HIGH-PERFORMANCE NETWORKS

Megaco/H.248 Protocol Implementation in OPNET

Spring 2002

Final Project Report

M.Riadul Mannan, Mahmood Riyadh, Shufang Wu
{mrmannan,mmriyadh,vswu}@cs.sfu.ca

Project Webpage
http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

Department of Computing Science
Simon Fraser University
8888 University Drive
Burnaby, BC, V5A 1S6
Canada

Date: April 20, 2002

Abstract

Currently, the best-known media gateway control protocols are Media Gateway Control Protocol (MGCP) and Megaco/H.248. MGCP was published as informational RFC 2705 and has been widely deployed. As an evolution of MGCP, Megaco/H.248 is expected to win wide industry acceptance as the official standard. The Internet Engineering Task Force (IETF) and International Telecommunication Union (ITU) have agreed to cooperate on a single unified protocol which is Megaco/H.248.

In our project, we have implemented an environment using OPNET to simulate Megaco/H.248. The simulation scenario has one Media Gateway Control (MGC) and two Media Gateways (MGs) connected to a local hub by point-to-point duplex link. We have implemented five Megaco/H.248 commands in order to simulate three basic call flows to demonstrate how Megaco/H.248 works. The data traffic between two MGs is simulated using Real-time Transport Protocol (RTP). The basic call flows are Successful MG Registration Procedure, Successful Call Setup Procedure and Successful Call Release Procedure. The details of these call flows are given in this report.

In the project, we practice software management skills and follow the SEI CMM Level 2 requirements to make the deliverables of our project repeatable. The main phases in the software cycle of our project are Concept Exploration, Requirement Analysis, Design and Implementation, Testing and finally, Integration.

We include the details of each phase and the simulation results of three basic call flows in this report. Finally, we give a problem summary and some possible future enhancements of our project.

Contents

Chapter 1 Introduction

1.1 Overview

Megaco/H.248 Protocol is a media gateway control protocol. Megaco stands for **Me**dia **Ga**teway **Co**ntrol protocol. It is the name from IETF and ITU-T calls it H.248.

Media gateway control protocols were established for the need of IP networks to interwork with traditional telephony systems and provide support for large-scale phone-to-phone deployments. Therefore, Megaco/H.248 is a solution for the traditional telephone network to transmit voice traffic over IP network.

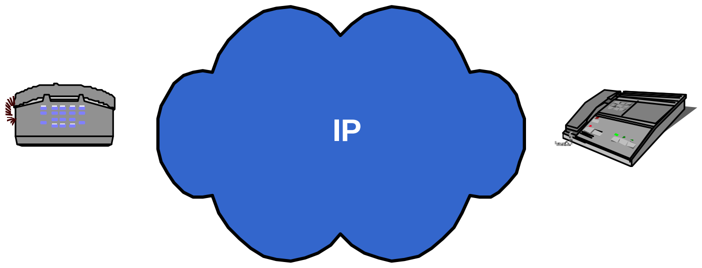


Figure 1.1 Telephones and IP Network.

While some other multimedia over IP protocols like Session Initiation Protocol (SIP) and H.323 are based on a peer-to-peer architecture, media gateway control protocols specify a master/slave architecture for decomposed gateways, in which MGC is the master serve r, and one or more MGs are the slave clients that behave like simple switches. One MGC can serve many MGs. Figure 1.2 [8] shows the operations in detail of Media Gateways and the mapping between IP and traditional Telephony networks.

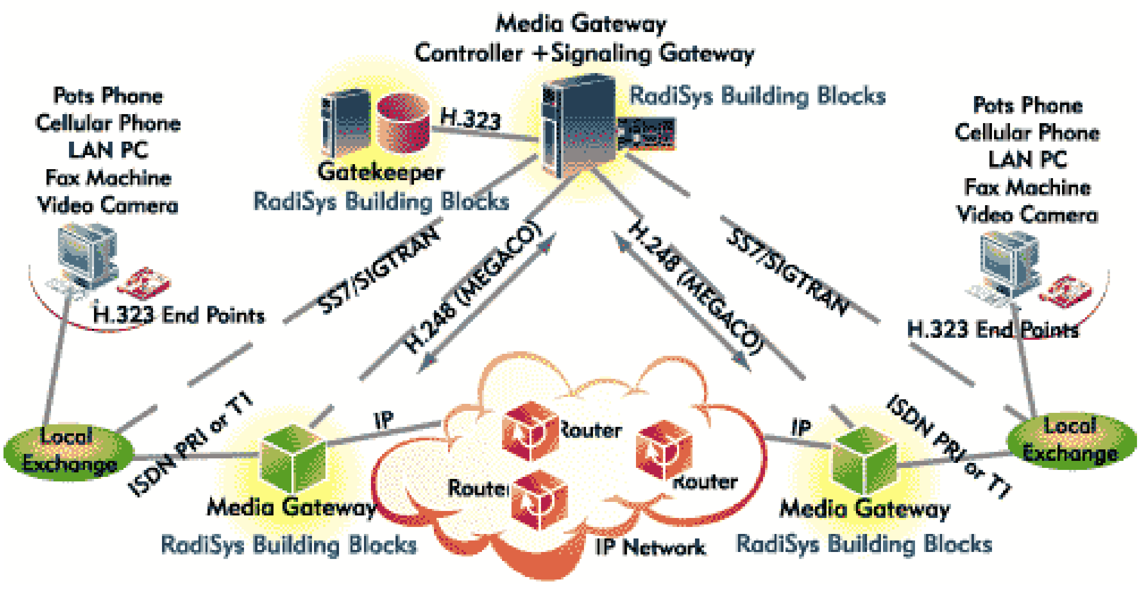


Figure 1.2 Media Gateway, solution for next generation telephony network [8].

Megaco/H.248 is not only attractive to important international standard organizations, but also to many industry giants like Cisco, Lucent, Nortel, Microsoft and Motorola. Cisco has implemented some products of previous versions to meet the needs of the market. Others are doing some research and development on it. Hopefully, Megaco/H.248 will be more and more important in both technology and market. Before we go into the details of our project, we give some background information on Media Gateways and their elements.

1.2 Background

1.2.1 Megaco/H.248

Megaco is the emerging standard that defines the interface between MG and MGC identified in the distributed Gateway Architecture proposed by ETSI (the European Telecommunications Standards Institute) project TIPHON (Telecommunications and Internet Protocol Harmonization Over Networks). The Megaco standard is the byproduct of the IETF and ITU 'joining hands' in effort to merge parallel development efforts.

1.2.2 Media Gateway Controller (MGC)

MGCs or “softswitches” are the foundation for next-generation networks offering the intelligence and reliability of the circuit-switched network with the speed and economy of the packet-switched network. As such, they must economically support both existing voice services like CLASS 4/5 features and evolving applications such as integrated access, VPN, Internet call waiting, click to dial, unified messaging, and enhanced roaming.

1.2.3 Media Gateway (MG)

MG implements the media mapping and transformation function identified in the decomposed Gateway architecture proposed by ETSI TIPHON and later adopted by IETF. It is the gateway between two different networks, e.g. IP and PSTN network and allows them to communicate.

1.2.4 Real-time Transport Protocol (RTP)

RTP is a Real-time Transport Protocol that provides end-to-end transport functions suitable for applications transmitting real-time data such as interactive audio, video or simulation data. It does not address resource reservation and does not guarantee quality-of-service for real-time services. The service is further augmented by Real-time Control Protocol (RTCP) to allow monitoring the delivery of data. It is a protocol framework that is not deliberately complete and therefore, a complete specification of RTP for a particular application will require one or more companion documents.

1.3 Project Goal

In this project, our main goal is to implement and simulate Megaco/H.248 [5] protocol in OPNET simulation environment and we explore the followings [9]:

1. Gain experience on the network research using OPNET.
2. Understand the current new technologies in High-performance Networking.
3. Gain experience in developing software project as well as software project management

We also implement the RTP protocol in order to send and receive data traffic between two MGs. We use a simple local area network scenario with two MGs and one MGC for the simulation. The system architecture of our project is as follows:

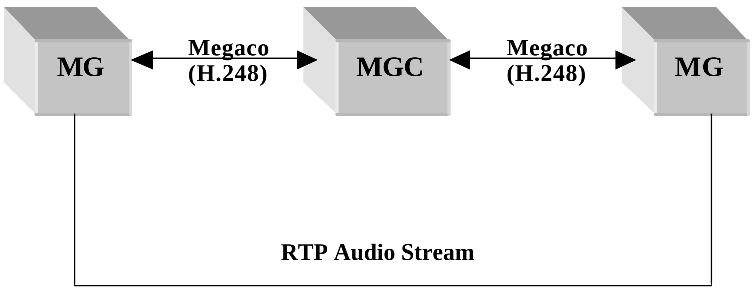


Figure 1.3 System Architecture.

In chapter 2, we give out the functional requirements of our project in detail. Based on the requirements, we describe the architecture of MG and MGC in chapter 3. Next, we show the basic call flow scenarios that we used for the simulation. In chapter 5, the implementation of our project in OPNET is explained. We present simulation results and analysis in chapter 6. Next, we discuss some problems that we have faced during and after the implementation and some possible solutions. Finally, we give a conclusion and possible future work directions.

Chapter 2 General Project Requirements

2.1 Overview

Here we highlight the basic requirements of our project. It is a good software engineering practice to make clear the requirements before implementation. In this report, we highlight only the main requirements. The complete specifications of the requirements are in [10], [11].

2.2 General Simulation Requirement

- OPNET Modeler 8.0.C (Build 1252) will be used.
- OPNET will be run on the host of payette.ensc.sfu.ca.

2.3 General Network Requirement

- Bus LAN (802.3) will be used.

2.4 General Transport Requirement

- ALF (Application Layer Framing) over UDP will be used between MG and MGC.
- RTP (Real-time Transport Protocol) over UDP will be used between MG and MG.

2.5 General Message Requirement

- ABNF text encoding of Megaco protocol will be used.
- Each message has:
  - ✓ A Protocol name
  - ✓ A Version number
  - ✓ An IP address of sender
  - ✓ An UDP port number of sender
  - ✓ One transaction.

2.6 General Transaction Requirement

- Each transaction has:
  - ✓ A TransactionID
  - ✓ One context.
- Each transaction will be presented by a pair of TransactionRequest and TransactionReply:
  - ✓ TransactionRequest is invoked by the sender
  - ✓ TransactionReply is invoked by the receiver
  - ✓ One pair of TransactionRequest and TransactionReply have the same TransactionID.
- TransactionID Requirement:

  - ✓ TransactionID is assigned by sender
  - ✓ TransactionID is unique within the scope of sender
  - ✓ The value of TransactionID is one from 1 to 99999, inclusively.

2.7 General Context Requirement

- Each context has:
  - ✓ A ContextID
  - ✓ One or two commands.
- ContextID Requirement:
  - ✓ ContextID is assigned by MG
  - ✓ ContextID is unique within the scope of MG
  - ✓ The value of ContextID is 1, "-" or "$".

2.8 General Command Requirement

- A command is either a request or a reply.
- The following commands will be used:
  - ✓ ServiceChange
  - ✓ Modify
  - ✓ Notify
  - ✓ Add
  - ✓ Subtract.

2.9 General MGC Requirement

- There will be only one MGC.
- The interface between MGC and MG is defined in System Interface Control Document (SICD) [12].

2.10 General MG Requirement

- There will be two MGs controlled by one MGC.
- Each MG has the following information of its MGC:
  - ✓ An IP address
  - ✓ An default UDP port number.
- The following signals will be used:
  - ✓ "cg/dt" for dial tone
  - ✓ "cg/rt" for ringing tone.
- The following events will be used:
  - ✓ "key/kd" for key down

    - ✓ "key/ku" for key up.
- The following TerminationIDs will be used:
    - ✓ "ui" for user interface termination
    - ✓ "at/hf" for handsfree of audio transducer termination
    - ✓ "tr" for RTP audio traffic.

2.11 General RTP Requirement

- Two Kinds of RTP Packet will be used:
    - ✓ RTP Packet
    - ✓ RTCP Packet.
- RTP Packet has:
    - ✓ A Fixed Length Header
    - ✓ RTP Payload.
- RTCP Packet has:
    - ✓ A Fixed Length Header
    - ✓ RTCP Packet Type.

In next section, we discuss the architecture of MGC and MG in detail.

Chapter 3 Architecture

3.1 Overview

MGC and MG are the two main components of our project. The architecture of these two components is based on Software Functional Specification (SWFS) of MGC, MG and RTP [13], [14], [15].

3.2 MGC Architecture

MGC has three main sub-components: (1) Message Analyzer (MA), (2) Message Sender (MS) and (3) Message Receiver (MR). The main control intelligence of MGC is located in message analyzer. Basically, it implements all the five commands that we have discussed in the earlier chapter. Upon receiving message from MG, first it extracts the message and interprets the commands and then, based on the received

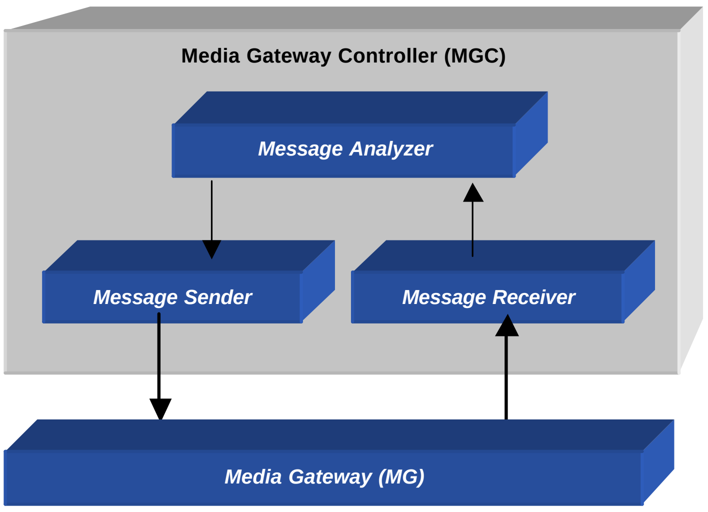


Figure 3.1 The architecture of MGC.

commands it takes actions accordingly. The communications between the sub-components are local function calls. The following are the responsibilities of each sub component.

Message Receiver is responsible for:

- *Receiving messages from MG*
- *Extracting required information from received messages*
- *Sending messages to MA.*

Message Analyzer is responsible for:

- *Receiving messages from MR*
- *Analyzing messages after receiving messages*
- *Sending messages to MS if needed.*

Message Sender is responsible for:

- *Receiving messages from MA*

- *Packaging required messages*
- *Sending messages to MG.*

3.3 MG Architecture

MG has four main sub-components: MG Processor, RTP, Audio Transducer and User Interface. The responsibilities of MG Processor are as follows:

- *Receiving messages from MGC or other MG*
- *Analyzing messages after receiving from MGC or MG*
- *Sending messages to MGC or MG*
- *Triggering audio transducer to start generating voice traffic*
- *Receiving statistical information from RTP*
- *Processing events, signals.*

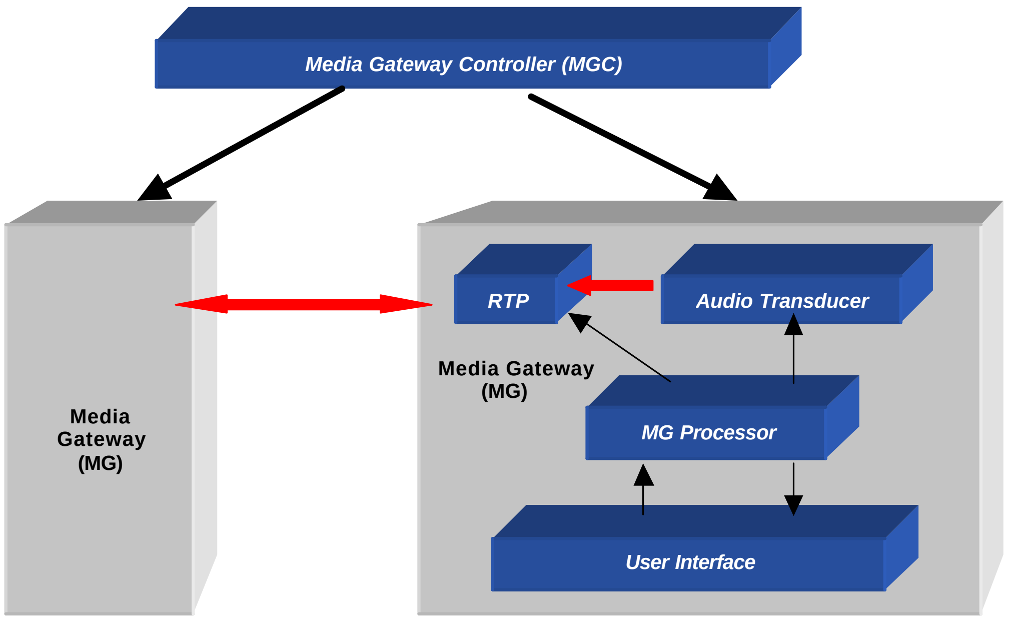


Figure 3.2 The architecture of MG.

Audio Transducer is responsible for generating the data traffic and RTP encodes the data traffic according to the encoding schemes specified in [2] and sends the RTP payloads to other MG. User interface provides the means for communication between user and the system.

Chapter 4 Call Flow Scenarios

4.1 Overview

In this project, we implement and simulate three call flow scenarios in Megaco IP Network. The scenarios are Successful MG Registration Procedure, Successful Call Setup Procedure and Successful Call Release Procedure. The following sections describe each call flow and the related parts in software specifications are attached as Appendix A, B, and C. Figure 4.1, Figure 4.2 and Figure 4.3 show the three call flows. In these figures, we describe the commands, main parameters and their values in the message.

4.2 Successful MG Registration Procedure

- MGC starts up before MG.
- After MG starts up, it sends a ServiceChange command to its MGC.
- The ServiceChange command has a ServiceChangeMethod of Restart.
- **Upon receipt of ServiceChange from MG successfully**:
    - MGC sends a positive response
    - The response has no ServiceChangeMethod
    - MGC sends MG a Modify command to ask MG to wait for Key Down event.
- **Upon receipt of Modify from MGC successfully**, MG sends a positive response.

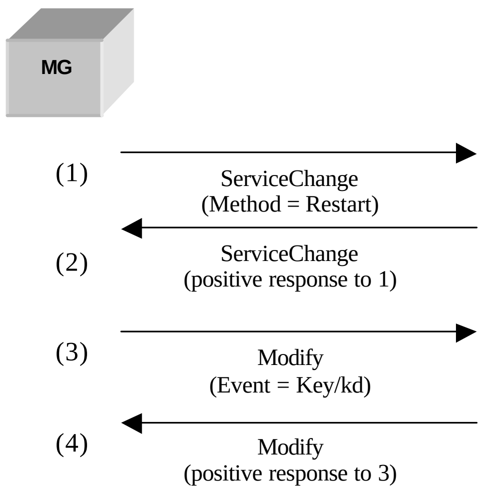


Figure 4.1 Call flow of Successful MG Registration Procedure.

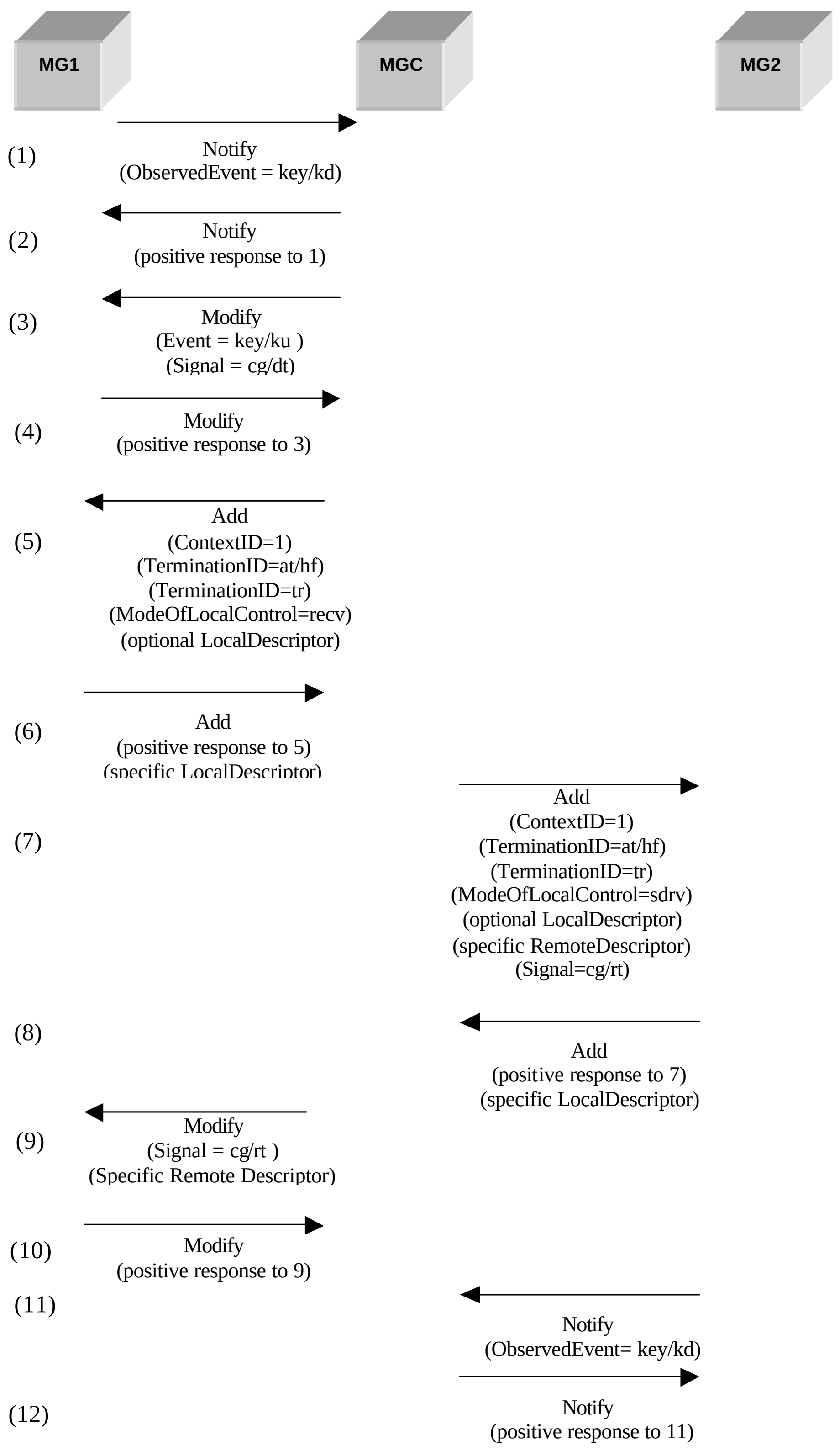
4.3 Successful Call Setup Procedure

- When MG1 detects Key Down event, it sends a Notify command to its MGC.
- The Notify command has an ObservedEventsDescriptor.
- **Upon receipt of Notify from MG1 successfully:**
    - MGC sends a positive response
    - MGC sends MG1 a Modify command

    - MGC asks MG1 to wait for Key Up event
    - MGC asks MG1 to generate a dial tone.
- **Upon receipt of Modify from MGC successfully**, MG1 sends a positive response.
- **Upon receipt of the above positive response from MG1 successfully:**
    - MGC sends MG1 an Add command
    - Specific ContextID is provided
    - Specific TerminationIDs are provided
    - Mode of LocalControlDescriptor is set to ReceiveOnly
    - Optional LocalDescriptor is provided.
- **Upon receipt of Add from MGC successfully:**
    - MG1 sends a positive response
    - Specific LocalDescriptor is provided.
- **Upon receipt of the above positive response from MG1 successfully:**
    - MGC sends MG2 an Add command
    - Mode of LocalControlDescriptor is set to Send/Receive
    - Specific RemoteDescriptor is provided
    - MGC asks MG2 to generate a ringing tone.
- **Upon receipt of Add from MGC successfully:**
    - MG2 sends a positive response
    - Specific LocalDescriptor is provided.
- **Upon receipt of the above positive response from MG2 successfully:**
    - MGC sends MG1 a Modify command
    - Specific RemoteDescriptor is provided
    - MGC asks MG1 to generate a ringing tone.
- **Upon receipt of Modify from MGC successfully:**
    - MG1 sends a positive response
    - When MG2 detects Key Down event, it sends a Notify command to its MGC
    - The Notify command has an ObservedEventsDescriptor.
- **Upon receipt of Notify from MG2 successfully:**
    - MGC sends a positive response
    - MGC sends MG2 a Modify command
    - MGC asks MG2 to wait for Key Up event
    - MGC asks MG2 to stop the ringing tone.
- **Upon receipt of Modify from MGC successfully**, MG2 sends a positive response.
- **Upon receipt of the above positive response from MG2 successfully:**
    - MGC sends MG1 a Modify command
    - MGC asks MG1 to stop the ring back

- Mode of LocalControlDescriptor is set to Send/Receive.

- **Upon receipt of Modify from MGC successfully**, MG1 sends a positive response.



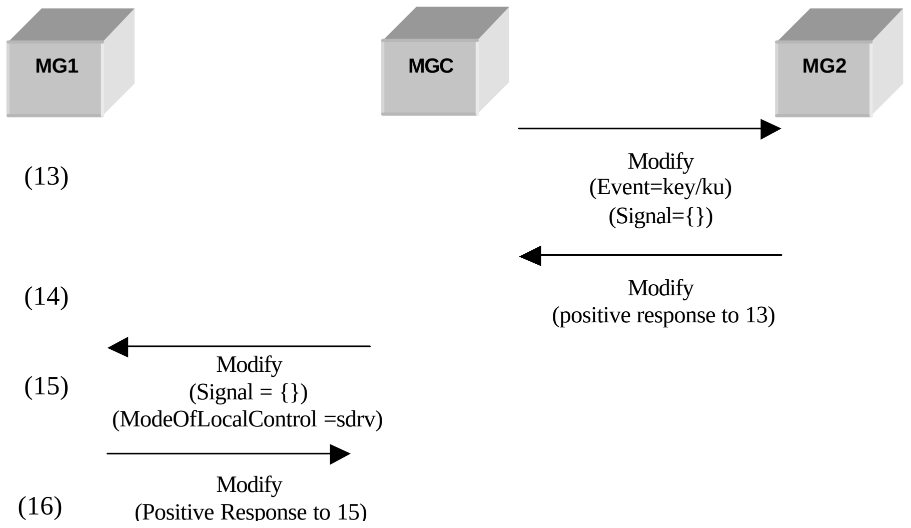


Figure 4.2 Call flow of Successful Call Setup Procedure.

4.4 Successful Call Release Procedure

- When MG1 detects Key Up event, it sends a Notify command to its MGC:
    - The Notify command has an ObservedEventsDescriptor.
- **Upon receipt of Notify from MG1 successfully:**
    - MGC sends a positive response
    - MGC sends MG1 a Subtract command:
        - Specific ContextID is provided
        - Specific TerminationIDs are provided.
    - **Upon receipt of Subtract from MGC successfully:**
        - MG1 sends a positive response
        - MGC sends a Subtract command to MG2:
            - Specific ContextID is provided
            - Specific TerminationIDs are provided.
    - **Upon receipt of Subtract from MGC successfully:**
        - MG2 sends a positive response
        - MGC sends MG1 a Modify command to ask MG1 to wait for Key Down event
    - **Upon receipt of Modify from MG C successfully:**
        - MG1 sends a positive response
        - MGC sends MG2 a Modify command to ask MG2 to wait for Key Down event.
    - **Upon receipt of Modify from MGC successfully**, MG2 sends a positive response.

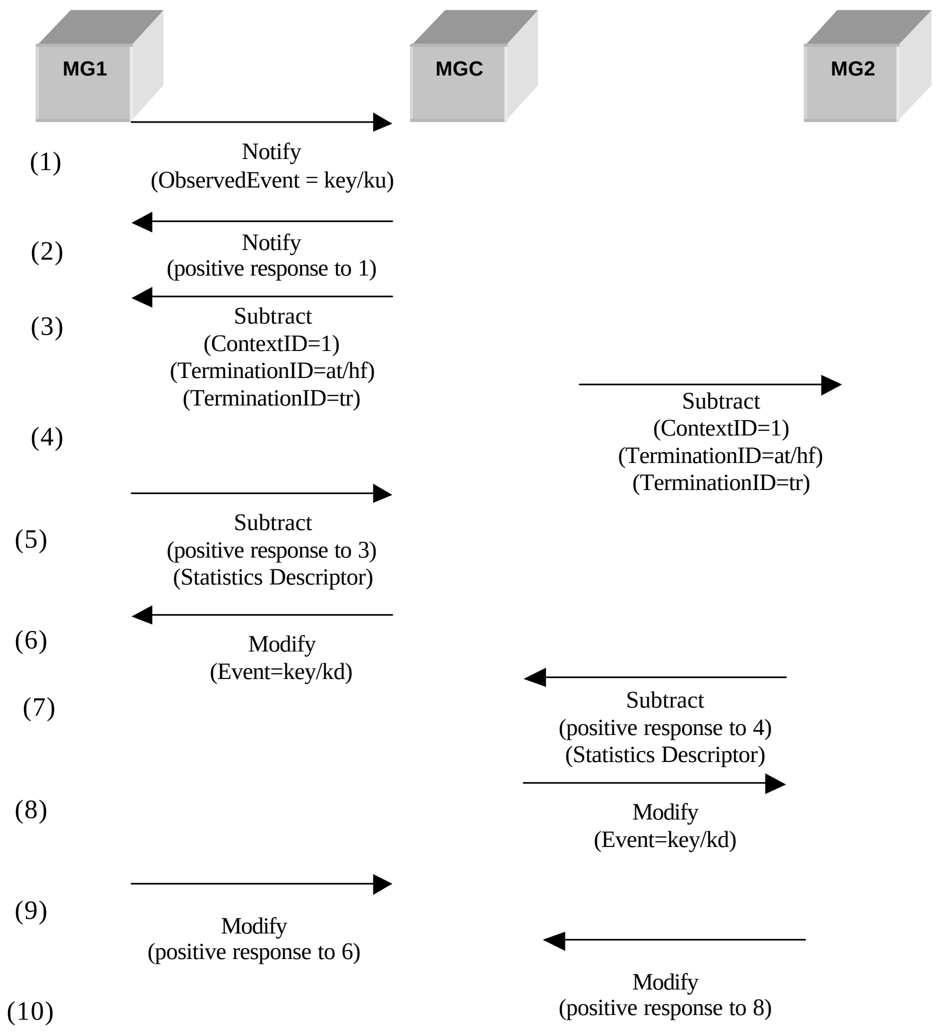


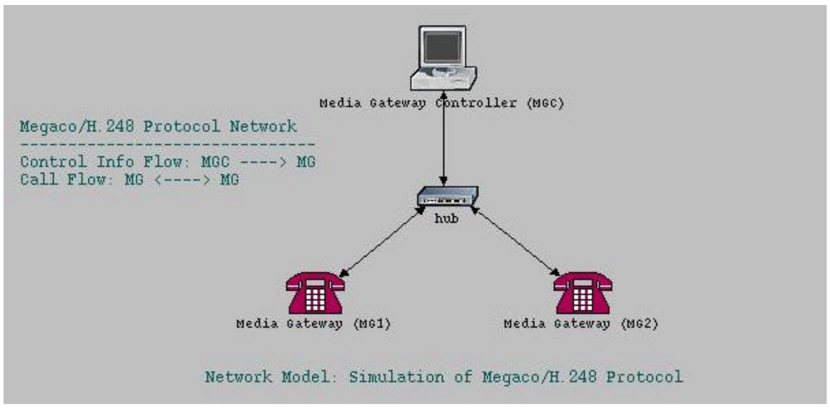
Figure 4.3 Call flow of Successful Call Release Procedure.

Chapter 5 Implementation in OPNET

5.1 Network Model and Node Models

Initially, we build a simple network model to simulate Megaco [5] protocol in the OPNET simulation environment based on the software design specification of MGC, MG and RTP [16], [17], [18]. Figure 5.1 illustrates the network model and simulation scenario in OPNET. It has three nodes connected to a local hub by direct point-to-point duplex link. This is a master-slave architecture where top node is the master node i.e. MGC and two other nodes are slaves i.e. MGs.

The protocol used between MGC and MG is Megaco/H.248 protocol as specified in the standards of [5] and [6]. RTP protocol is used for the communication between two MGs. The standard is [1].


Media Gateway Controller (MGC)
Megaco/H.248 Protocol Network

Control Info Flow: MGC ----> MG
Call Flow: MG <----> MG
hub
Media Gateway (MG1)
Media Gateway (MG2)
Network Model: Simulation of Megaco/H.248 Protocol

Figure 5.1 Network Model.

Figures 5.2 and 5.3 illustrate the node models of MG and MGC in OPNET simulation environment. Each model has only one layer, application layer, and one transmitter and one receiver. The application layers of MG and MGC node models contain MG process model and MGC process model respectively.

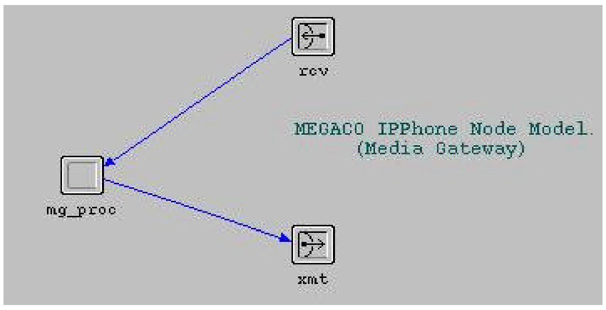


Figure 5.2 MG Node Model.

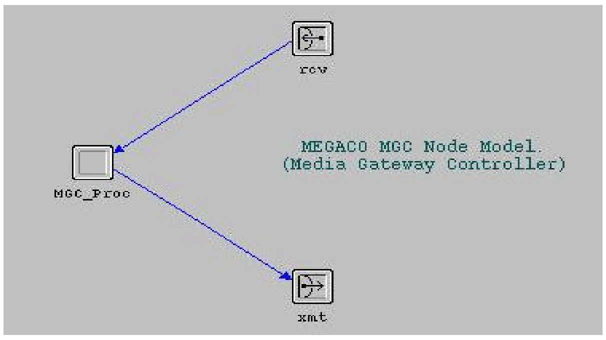


Figure 5.3 MGC Node Model.

The hub node model is shown in Figure 5.3. It has three transmitters and three receivers and one processor. The processor retrieves the destination address from the incoming packet. Based on the destination address, it sends the packet to appropriate transmitter. We define the transmitters and receivers for each MG and MGC, e.g., MGC_recv and MGC_xmt are connected to MGC receiver and transmitter respectively.

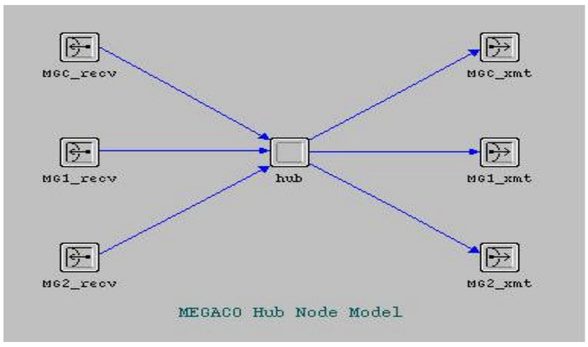


Figure 5.4 Hub Node Model.

5.2 MGC Process Model

The MGC process model has three states, init, idle and process. Init state initializes the MGC when it is started for the first time and then, moves to the idle state where it waits for the arrival of Megaco packet. Upon the arrival of a Megaco packet, it moves to process state where all the control intelligence of MGC is

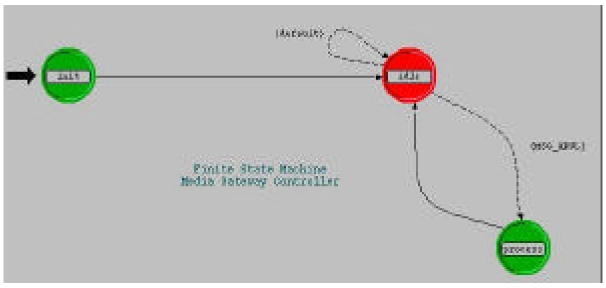


Figure 5.5 MGC Process Model.

located. Here, it unpacks the packets and then, parses it accordingly. Afterwards, it interprets the command type and takes necessary steps.

5.3 MG Process Model

The MG process model is a little bit more complex than the MGC process model as it has six states, i.e., init, idle, MG_Proc, RTP_Proc, MG_Send and MG_Recv. Init state initializes the MG and after the initialization, it sends a ServiceChange request to MGC in order to register itself. After that, it moves on to idle state where it waits for the acknowledgement from MGC for ensuring the acceptance of registration request. After getting the registration acknowledgement, it waits for further commands from MGC and listens for events to occur, e.g., key up, key down events. It also waits for RTP packets when the connection is established between two MGs. MG_Proc is responsible for interpreting the Megaco commands received from MGC and therefore, taking actions accordingly. When it detects a key down event, it notifies MGC about the

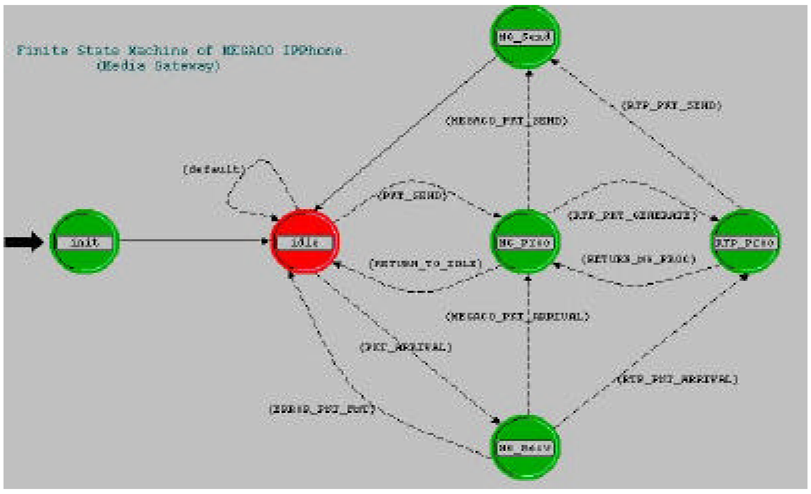


Figure 5.6 MG Process Model.

event and sets up the connection according to the instructions from MGC. When the connection is established successfully, control goes to RTP_Proc, which is responsible for generating data traffic between two MGs. This process builds the RTP packets that carry the RTP payload according to the standard RTP packet format specified in [1]. MG_Send is responsible for sending Megaco and RTP packets to the output stream using the underlying transport mechanism. MG_Recv extracts a packet from the input stream, when there is packet arrival interrupt encountered while it is in the idle state. It retrieves the packet format and the control is transferred to either MG_Proc or RTP_Proc depending on the packet sender. When MG_Proc detects a key up event, a Notify command is sent to MGC to terminate the connection that has been established after key down event between two MGs. Then MG receives Subtract command from MGC and finally, MG_Proc tears down the connection.

Chapter 6 Simulation and Results

The complete simulation results are as follows. The results are exactly what are explained in Chapter 4 and Appendix A, B and C.

Messages in Successful MG1 (IP address: 172.16.0.2) Registration Procedure:

[1] , [2], [3], [7].

Messages in Successful MG2 (IP address: 172.16.0.3) Registration Procedure:

[4] , [5], [6], [8].

Messages in Successful Call Setup Procedure:

From [9] to [24].

Messages in Successful Call Release Procedure:

From [25] to [34].

Also, between message [29] and [30], information of statistics of MG2 is shown.

Information of statistics of MG3 is shown between message [31] and [32].

```
|----------------------------------------------------------------------------|
|  Simulation Completed - Collating Results.                                 |
|  Events: Total (1944), Average Speed (7892 events/sec.)                    |
|  Time: Elapsed (1 sec.), Simulated (1 hr. 40 min. 0 sec.)                  |
|  Simulation Log: 1 entries                                                 |
|----------------------------------------------------------------------------|
|----------------------------------------------------------------------------|
|  ||||  |||||  ||   || ||||| |||||||| Network Simulation of:                |
| ||  || || ||| |||  || ||       ||    megaco_project_v4-scenario1           |
| ||  || |||||  |||| || |||||    ||                                          |
| ||  || ||     || |||| ||       ||    Simulation and Model Library Copyright |
| ||  || ||     ||  ||| ||       ||    1987-2002 by OPNET Technologies, Inc.  |
|  ||||  ||     ||   || |||||    ||    as a part of OPNET Release 8.0.C       |
|----------------------------------------------------------------------------|
|       O P T I M U M      N E T W O R K      P E R F O R M A N C E          |
|----------------------------------------------------------------------------|
|   OPNET Technologies, Inc. / 7255 Woodmont Av. / Bethesda, MD 20814, USA   |
|               TEL: +1.240.497.3000 / FAX: +1.240.497.3001                  |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Reading network model (megaco_project_v4-scenario1)                       |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Reading result collection file: (megaco_project_v4-scenario1)             |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Verifying model consistency.                                              |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Rebuilding scenario model library.                                        |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Loading scenario model library.                                           |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Counting objects.                                                         |
|----------------------------------------------------------------------------|

|----------------------------------------------------------------------------|
|  Allocating objects.                                                         |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Allocating objects.                                                         |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Allocating attributes.                                                      |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Installing subnetworks and nodes.                                           |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Installing links.                                                           |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Constructing model hierarchy.                                               |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Assigning object identification.                                            |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Creating node contents.                                                     |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Attaching links to nodes.                                                   |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Preparing for results collection.                                           |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Releasing Model Memory.                                                     |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Opening results file.                                                       |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Final initializations for all objects.                                      |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Initializing results.                                                       |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Progress: Time (0 sec.), Events (0)                                         |
|  Speed: Average (0 events/sec.), Current (0 events/sec.)                     |
|  Time: Elapsed (0 sec.)                                                      |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
| [ 1] MGC received the following message:                                     |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Transaction = 1005 {
Context = - {
ServiceChange = ROOT {
Services {
Method=Restart,
ServiceChangeAddress=5555,
Profile=IPPhone/1
}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 2] MGC sent MG1 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Reply = 1005 {
Context = - {
ServiceChange = Root {
Services {
ServiceChangeAddress=5555,
Profile=IPPhone/1
}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 3] MGC sent MG1 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 1 {
Context = - {
Modify = ui {
Events = 1 {key/kd}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 4] MGC received the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Transaction = 2005 {
Context = - {
ServiceChange = ROOT {
Services {
Method=Restart,
ServiceChangeAddress=5555,
Profile=IPPhone/1
}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 5] MGC sent MG2 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Reply = 2005 {
Context = - {
ServiceChange = Root {
Services {
ServiceChangeAddress=5555,
Profile=IPPhone/1
}
}
}
}
```

```
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 6] MGC sent MG2 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 2 {
Context = - {
Modify = ui {
Events = 2 {key/kd}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 7] MGC received the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 1 {
Context = - {
Modify = ui
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 8] MGC received the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Reply = 2 {
Context = - {
Modify = ui
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [ 9] MGC received the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Transaction = 1006 {
Context = - {
Notify = ui {
ObservedEvents = 1 {
20020419T827900:key/kd}
}
}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [10] MGC sent MG1 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Reply = 1006 {
Context = - {
Notify = ui}
}
|---------------------------------------------------------------------------|


|---------------------------------------------------------------------------|
| [11] MGC sent MG1 the following message:                                  |
|---------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
```

```
Transaction = 3 {
Context = - {
Modify = ui {
Events = 3 {key/ku}
}
,
Modify = at/hf {
Signals {cg/dt}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [12] MGC received the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 3 {
Context = - {
Modify = ui,
Modify = at/hf}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [13] MGC sent MG1 the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 4 {
Context = 1 {
Add = at/hf,
Add = tr {
Media {
LocalControl {
Mode=ReceiveOnly
}
,
Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 0
v=0
c=IN IP4 $
m=audio $ RTP/AVP 8
}
}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [14] MGC received the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 4 {
Context = 1 {
Add = -,
Add = tr {
Media {
Local {
v=0
c=IN IP4 172.16.0.2
m=audio 2222 RTP/AVP 0
}
}
}
}
```

```
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [15] MGC sent MG2 the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 5 {
Context = 1 {
Add = at/hf {
Signals {cg/rt}
}
,
Add = tr {
Media {
LocalControl {
Mode=SendReceive
}
,
Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 0
}
,
Remote {
v=0
c=IN IP4 172.16.0.2
m=audio 2222 RTP/AVP 0
}
}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [16] MGC received the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Reply = 5 {
Context = 1 {
Add = at/hf,
Add = tr {
Media {
Local {
v=0
c=IN IP4 172.16.0.3
m=audio 2222 RTP/AVP 0
}
}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [17] MGC sent MG1 the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 6 {
Context = 1 {
Modify = at/hf {
Signals {cg/rt}
}
,
Modify = tr {
Media {
```

```
Remote {
v=0
c=IN IP4 172.16.0.3
m=audio 2222 RTP/AVP 0
}
}
}
}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [18] MGC received the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 6 {
Context = 1 {
Modify = at/hf,
Modify = tr}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [19] MGC received the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Transaction = 2006 {
Context = 1 {
Notify = ui {
ObservedEvents = 2 {
20020419T827900:key/kd}
}
}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [20] MGC sent MG2 the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Reply = 2006 {
Context = - {
Notify = ui}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [21] MGC sent MG2 the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 7 {
Context = - {
Modify = ui {
Events = 4 {key/ku}
}
,
Modify = at/hf {
Signals {}
}
}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [22] MGC received the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Reply = 7 {
Context = 1 {
Modify = ui,
Modify = at/hf}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [23] MGC sent MG1 the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 8 {
Context = 1 {
Modify = at/hf {
Signals {}
}
,
Modify = tr {
Media {
LocalControl {
Mode=SendReceive
}
}
}
}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [24] MGC received the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 8 {
Context = 1 {
Modify = at/hf,
Modify = tr}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [25] MGC received the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Transaction = 2007 {
Context = 1 {
Notify = ui {
ObservedEvents = 4 {
20020419T827900:key/ku}
}
}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [26] MGC sent MG2 the following message:                                 |
|--------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Reply = 2007 {
Context = - {
Notify = ui}
}
|--------------------------------------------------------------------------|


|--------------------------------------------------------------------------|
| [27] MGC sent MG2 the following message:                                 |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 9 {
Context = 1 {
Subtract = at/hf,
Subtract = tr}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [28] MGC sent MG1 the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 10 {
Context = 1 {
Subtract = at/hf,
Subtract = tr}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [29] MGC received the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Reply = 9 {
Context = 1 {
Subtract = at/hf,
Subtract = tr {
Statistics {
rtp/ps=50,
rtp/pr=50,
rtp/pl=0,
rtp/jit=0,
rtp/delay=0}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
|  Module (51), (top.MGC.MGC_Proc)                                            |
|  From procedure: megaco_mgc_v6 () [process enter execs]                     |
|  Statistics received by MGC:                                                |
|  Number of packets sent by MG[IP:172.16.0.3] is 50                          |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
|  Module (51), (top.MGC.MGC_Proc)                                            |
|  From procedure: megaco_mgc_v6 () [process enter execs]                     |
|  Statistics received by MGC:                                                |
|  Number of packets received by MG[IP:172.16.0.3] is 50                      |
|-----------------------------------------------------------------------------|

|-----------------------------------------------------------------------------|
| [30] MGC sent MG2 the following message:                                    |
|-----------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 11 {
Context = - {
Modify = ui {
Events = 5 {key/kd}
}
}
}
|-----------------------------------------------------------------------------|


|-----------------------------------------------------------------------------|
| [31] MGC received the following message:                                    |
|------------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 10 {
Context = 1 {
Subtract = at/hf,
Subtract = tr {
Statistics {
rtp/ps=50,
rtp/pr=50,
rtp/pl=0,
rtp/jit=0,
rtp/delay=0}
}
}
}
|------------------------------------------------------------------------------|


|------------------------------------------------------------------------------|
|  Module (51), (top.MGC.MGC_Proc)                                             |
|  From procedure: megaco_mgc_v6 () [process enter execs]                      |
|  Statistics received by MGC:                                                 |
|  Number of packets sent by MG[IP:172.16.0.2] is 50                           |
|------------------------------------------------------------------------------|

|------------------------------------------------------------------------------|
|  Module (51), (top.MGC.MGC_Proc)                                             |
|  From procedure: megaco_mgc_v6 () [process enter execs]                      |
|  Statistics received by MGC:                                                 |
|  Number of packets received by MG[IP:172.16.0.2] is 50                       |
|------------------------------------------------------------------------------|

|------------------------------------------------------------------------------|
| [32] MGC sent MG1 the following message:                                     |
|------------------------------------------------------------------------------|
MEGACO/1 [172.16.0.1]:2944
Transaction = 12 {
Context = - {
Modify = ui {
Events = 6 {key/kd}
}
}
}
|------------------------------------------------------------------------------|


|------------------------------------------------------------------------------|
| [33] MGC received the following message:                                     |
|------------------------------------------------------------------------------|
MEGACO/1 [172.16.0.2]:5555
Reply = 12 {
Context = - {
Modify = ui
}
}
|------------------------------------------------------------------------------|


|------------------------------------------------------------------------------|
| [34] MGC received the following message:                                     |
|------------------------------------------------------------------------------|
MEGACO/1 [172.16.0.3]:5555
Reply = 11 {
Context = - {
Modify = ui
}
}
|------------------------------------------------------------------------------|
|--------------------------------------------------------------------------|
|  Simulation Completed - Collating Results.                               |
|  Events: Total (1944), Average Speed (7206 events/sec.)                  |
|  Time: Elapsed (1 sec.), Simulated (1 hr. 40 min. 0 sec.)                |
|  Simulation Log: 1 entries                                               |
|--------------------------------------------------------------------------|
```

Chapter 7   Problem Summary and Solutions

7.1 Interfacing with standard node modules

Initially, we thought of implementing Megaco protocol based on the layered architecture. Our idea was to write the application layer and interface it with the standard UDP layer provided by OPNET using Interface Control Information Package. In order to do so, we tried to get ideas from codes that exist in the OPNET and wrote functions according to our need. But, it turned out to be difficult to implement the protocol using all the layers. The original network model is shown in Figure 7.1. In this network model, we used standard Ethernet hub and link models. Although we have changed the architecture, our project goal has not been

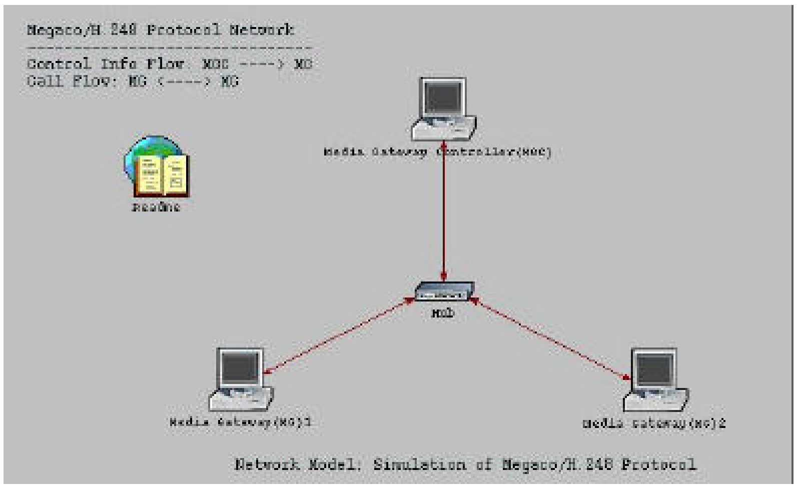

Megaco/H.248 Protocol Network
Control Info Flow: MGC ----> MG
Call Flow: MG <----> MG
Media Gateway Controller(MGC)
ReadMe
Hub
Media Gateway(MG)1
Media Gateway(MG)2
Network Model: Simulation of Megaco/H.248 Protocol

Figure 7.1 Initial Network Model.

changed for simulating the Megaco/H.248 protocol. Figure 7.2 describes the original node model that we have tried to implement in OPNET. It consists of six layers. The application layer contains all the applications that we have implemented in OPNET such as MGC, MG and RTP. We do not see RTP as a separate application rather it is integrated within the MG application. The application layer needs to communicate with lower layer transport mechanism in order to send and receive packets in the IP network. We use the UDP protocol as the underlying transport mechanism in our old network model. However, as mentioned earlier, we could not use this node model to simulate our network model in the OPNET environment due to some internal technical difficulties of OPNET during the simulation. We tried to fix the problem for several days and nights. The problems could be the interface between two layers or understanding the simulation attributes of different layers or the OPNET itself. Due to time constraint and after consulting with our professor, we decided to switch the architecture of our node model to a simple one where it has only application layer as we mentioned in section 5.1. We made small changes in our process model to incorporate the new network model.

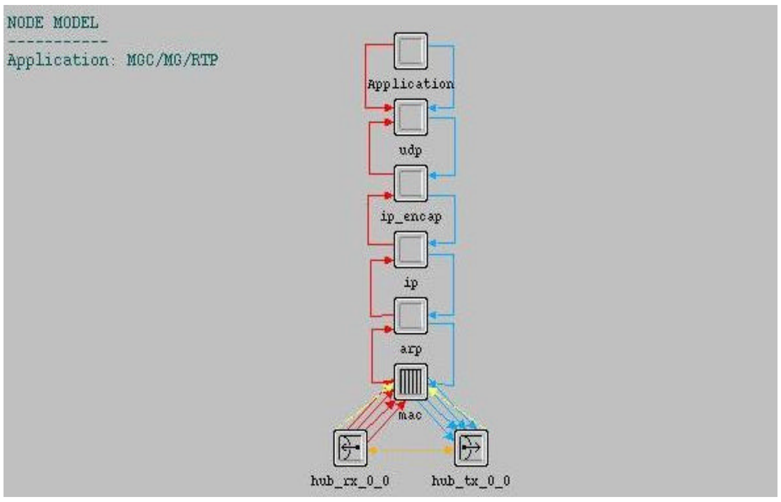


Figure 7.2 Initial Node Model.

7.2 Memory Management

Special care has to be taken for dynamic allocation/de-allocation of memory since all the codes in OPNET have to be written by C/C++ and they don't provide automatic garbage collection like Java. We encountered bus errors and segmentation faults.

To solve the problem, we tried to keep number of memory allocation/de-allocation as minimum as possible especially for messages that are sent/received to and from one node to other. We have also declared fixed array size wherever possible making sure the code is as flexible as possible.

7.3 OPNET Interrupt

Interrupt plays a very important part in OPNET in the Finite State Machine (FSM) design. Interrupt is a must in order to check the transition condition from an unforced state.

In the FSM of MG we have used self interrupt to move from idle state to MG_Proc state in order to send ServiceChange and Notify commands to MGC as these are not dependent on any packet arrival. To accomplish this, we set the <intrpt interval> attribute of MG process model to 20, which schedules an interrupt for the root process of the processor module after every 20 seconds.

7.4 Transition Condition

It requires special care to ensure that, more than one transition conditions don't become true in a specific state, which has multiple transition conditions set up with other states.

Chapter 8 Conclusion and Future Work

Conclusion

During the project, we have gained some experience on the network research using OPNET. We know how to set up a simple network model to do research in OPNET. Also during the working, we found some problems in both OPNET and our codes. In resolving the problems, we have learnt a lot.

Understand the current new technologies in High-performance Networking. Gain experience in developing software project as well as software project management. We make our project planned first, well managed in a professional method. Actually our project conforms to CMM Level 2(http://www.sei.cmu.edu/cmm/). We think it is very helpful for coding.

As for the Megaco/H.248, we understand its usefulness much more during the project. It is possible to implement it as a product.

Future Work

In future we would like to do the followings:

- Implement other three Megaco/H.248 commands
- Implement full featured RTP
- Multiple call scenarios
- Using layer architecture to implement Megaca/H.248

Acknowledgement

We would like to give thanks to Professor Ljiljana Trajkovic for giving nice introduction of latest knowledge on this field. Based on what we have learnt in class, we can do our project with enough background and be easy to understand lot of concepts. Also, we would like to give thanks to both of our TAs for helping us during the project work.

References

1. Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, January 1996: http://www.ietf.org/rfc/rfc1889.txt (accessed in February 2002).

2. Schulzrinne, H., "RTP Profile for Audio and Video Audio and Video Conferences with Minimal Control," RFC 1890, January 1996: http://www.ietf.org/rfc/rfc1890.txt (accessed in February 2002).

3. Handley, M. and V. Jacobson, "SDP: Session Description Protocol," RFC 2327, April 1998: http://www.ietf.org/rfc/rfc2327.txt (accessed in February 2002).

4. Greene, N., Ramalho, M. and B. Rosen, "Media Gateway control protocol architecture and requirements," RFC 2805, April 1999: http://www.ietf.org/rfc/rfc2805.txt (accessed in February 2002).

5. Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. and J. Segers, "Megaco Protocol Version 1.0," RFC 3015, November 2000: http://www.ietf.org/rfc/rfc3015.txt (accessed in February 2002).

6. Blatherwick, P., Bell, R. and P. Holland, "Megaco IP Phone Media Gateway Application Profile," RFC 3054, January 2001: http://www.ietf.org/rfc/rfc3054.txt (accessed in February 2002).

7. Brown, M. & P. Blatherwick, “ITU-T SG16, H.248 Annex G: User Interface Elements and Actions Packages,” November 2000: ftp://standards.nortelnetworks.com/megaco/docs/Approved/H248_G_PDF.zip (accessed in February 2002).

8. “Radisys Building Blocks for Media Gateway solutions,” http://www.radisys.com/files/media_gateway.swf

9. “Project Plan (PP),”: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

10. “System Functional Specification (SFS 001),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

11. “System Functional Specification of Custom RTP (SFS 002),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

12. “System Interface Control Document (SICD),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

13. “Software Functional Specification of MG (SWFS 001),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

14. “Software Functional Specification of MGC (SWFS 002),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

15. “Software Functional Specification of Custom RTP (SWFS 003),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

16. “Software Design Specification of MG (SWDS 001),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

17. “Software Design Specification of MGC (SWDS 002),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

18. “Software Design Specification of Custom RTP (SWDS 003),” March 2002: http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

Appendix A

Successful MG Registration Procedure

MA = Message Analyzer
MS = Message Sender
MR = Message Receiver

- **Upon receipt of ServiceChange from MG successfully,**
- MA creates a MG with the following information:
  - IP address
  - Service address
  - Termination ID of "ui"
  - Termination ID of "at/hf"
  - Termination ID of "tr".
- MA initializes all Termination IDs of the MG to free status.
- MA creates a new Transaction ID.
- MA creates a new Request ID.
- MA sets the status of "ui" to busy for event.
- MA sends MS a Modify message with following parameters:
  - Number of commands with the value 1
  - Termination ID of "ui"
  - A sign value shows that Events Descriptor exists
  - Package Name of Key Down event
  - Event ID of Key Down event.
- MA starts a timer related to the Transaction ID.
- **Upon receipt of Modify from MG successfully,**
  - MR sends MA a Modify message with following parameters:
    - Number of commands with the value 1.
  - MA verifies whether the following MG exists:
    - IP address in the received message
    - UDP port number in the received message.
  - If the MG exists, MA verifies the Transaction ID:
    - If the Transaction ID exists,
      - MA cancels the related timer
      - MA deletes the Transaction ID.

Appendix B

Successful Call Setup Procedure

Upon receipt of Notify from MG1 successfully,

- If the MG1 exists, MA verifies whether the Request ID exists.
- If the Request ID exists,
- Delete the Request ID.
- MA sets the status of "ui" of MG1 to free for event.
- MA adds the following information to the MG1,
- Key down event happens.
- MA creates a new Transaction ID.
- MA creates a new Request ID.
- MA sets the status of "ui" of MG1 to busy for event.
- MA sets the status of "at/hf" of MG1 to busy for signal.
- MA sends MG1 through MS a Modify message with following parameters,
- Number of commands with the value 2.
- First Termination ID of "ui".
- A sign value shows that Events Descriptor exists for first Termination ID.
- Package Name of Key up event.
- Event ID of Key up event.
- Second Termination ID of "at/hf",
- A sign value shows that Signals Descriptor exists for second Termination ID.
- A sign shows signal exists.
- Package Name of dial tone signal.
- Signal ID of dial tone signal.
- MA starts a timer related to the Transaction ID.

Upon receipt of Modify from MG1 successfully,

- MR sends MA a Modify message with following parameters,
  - Number of commands with the value 2.
- MA creates a new Transaction ID.
- MA sets the Context ID of "tr" of MG1 to "1".
- MA sets the Context ID of "at/hf" of MG1 to "1".
- MA sends MG1 through MS an Add message with following parameters:
  - Context ID of "1"
  - First Termination ID of "at/hf"
  - Second Termination ID of "tr"
  - Mode value of "ReceiveOnly"
  - A sign value shows no Remote Descriptor
  - Others refer to SWFS-002-8.4.
- MA starts a timer related to the Transaction ID.

Upon receipt of Add from MG1 successfully,

- MA sets the following attributes of "tr" of MG1:
  - IP address
  - UDP port number
  - Codec value
  - MA creates a new Transaction ID
  - MA sets the Context ID of "tr" of MG2 to "1"
  - MA sets the Context ID of "at/hf" of MG2 to "1"
  - MA sets the status of "at/hf" of MG2 to busy for signal
  - MA sends MG2 through MS an Add message with following parameters
  - Context ID of "1"

- First Termination ID of "at/hf".
- Second Termination ID of "tr":
  - Mode value of "SendReceive"
  - A sign value shows a Remote Descriptor
  - Package Name of ringing tone signal
  - Signal ID of ringing tone signal
  - IP address of "tr" of MG1
  - UDP port number of "tr" of MG1
  - Codec value of "tr" of MG1
  - MA starts a timer related to the Transaction ID.

Upon receipt of Add from MG2 successfully,

- MA sets the following attributes of "tr" of MG2:
  - IP address
  - UDP port number
  - Codec value
  - MA creates a new Transaction ID
  - Number of commands with the value 2.
- First Termination ID of "at/hf":
  - A sign value shows a Signals Descriptor for first Termination ID
  - A sign shows signal exists
  - Package Name of ringing tone signal
  - Signal ID of ringing tone signal.
- Second Termination ID of "tr":
  - A sign value shows a Remote Descriptor for second Termination ID
  - IP address of "tr" of MG2
  - UDP port number of "tr" of MG2
  - Codec value of "tr" of MG2.

Upon receipt of Modify from MG1 successfully:

- Number of commands with the value 2.

Upon receipt of Notify from MG2 successfully:

- If the MG2 exists, MA verifies whether the Request ID exists
- If the Request ID exists,
- MA sets the status of "ui" of MG2 to busy for event
- MA sets the status of "at/hf" of MG2 to free for signal
- MA sends MG2 through MS a Modify message with following parameters
- A sign value shows no signal
- Others refer to SWFS-002-8.3

Upon receipt of Modify from MG2 successfully:

- MA creates a new Transaction ID
- MA sets the status of "at/hf" of MG1 to free for signal
- A sign value shows no signal
- A sign value shows a Local Control Descriptor for second Termination ID
- Mode value of "SendReceive".

Appendix C

Successful Call Release Procedure

- **Upon receipt of Notify from MG successfully:**
- If the MG exists, MA verifies whether the Request ID exists.
- If the Request ID exists:
    - Delete the Request ID
    - MA sets the status of "ui" to free for event
    - MA adds the following information to the MG
    - Key up eve nt happens.
- MA initializes all Termination IDs of the MG to free status.
- MA creates a new Transaction ID for the MG.
- MA starts a timer related to the Transaction ID.
- MA sets all Termination IDs of the other MG to free status.
- MA creates another new Transaction ID for the other MG.
- MA starts another timer related to another Transaction ID.
- **Upon receipt of Subtract from MG successfully:**
    - Release the call and store the statistics