OPNET Implementation of Megaco/H.248 Protocol

Shufang Wu, Mahmood Riyadh, Riadul Mannan, and Ljiljana Trajkovic
Simon Fraser University
Vancouver, British Columbia
Canada V5A 1S6
E-mail: {vswu, mmriyadh, mrmannan, ljilja}@cs.sfu.ca
http://www.sfu.ca/~vswu/courses/CMPT885/project.htm

Abstract
The best-known media gateway control protocols are currently Media Gateway Control Protocol (MGCP) and Megaco/H.248. MGCP, described as informational RFC 2705, has been widely deployed. Megaco/H.248 has evolved from MGCP. It is a single unified protocol that resulted from the cooperation between the Internet Engineering Task Force (IETF) and the International Telecommunication Union (ITU). Megaco/H.248 is expected to win wide industry acceptance as the official standard.

In this paper, we describe OPNET implementation of Megaco/H.248. Our simulation scenario employs one Media Gateway Control (MGC) and two Media Gateways (MGs) connected to a local hub by point-to-point duplex links. We have implemented five Megaco/H.248 commands in order to simulate three basic call flows and demonstrate Megaco/H.248 functionality. Data traffic between two MGs is simulated using the Real-Time Transport Protocol (RTP). We describe the OPNET implementation details of the three basic call flows: Successful MG Registration Procedure, Successful Call Setup Procedure, and Successful Call Release Procedure. We also present simulation results and discuss possible future enhancements of our implementation.

1. Introduction

Megaco/H.248 protocol [3] is a media gateway control protocol. Megaco, named by IETF, stands for MEdia GAteway COntrol protocol. ITU-T calls it H.248. Megaco/H.248 is the emerging standard that defines the interface between Media Gateway Control (MGC) and Media Gateways (MGs) identified in the distributed Gateway Architecture proposed by ETSI (the European Telecommunications Standards Institute) project TIPHON (Telecommunications and Internet Protocol Harmonization Over Networks) [9]. The Megaco standard is the byproduct of the IETF and ITU "joining hands" in effort to merge parallel development efforts.

Megaco/H.248 is not only attractive to international standard organizations, but also to industrial companies such as Cisco, Lucent, Nortel, Microsoft, and Motorola. For example, to meet the market demands, Cisco has implemented several network products employing Media Gateway Control Protocol (MGCP). It is expected that Megaco/H.248 will become increasingly important in the networking market.

1.1. Media gateway and its elements

Media gateway control protocols were introduced to inter-network IP and traditional telephony systems and to provide support for large-scale end-to-end deployments. Hence, Megaco/H.248 enables traditional telephone networks to transmit voice traffic over IP. While other multimedia over IP protocols, such as 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 the master/slave architecture, MGC is the master server and MGs are the slave clients that behave as simple switches. One MGC can serve multiple MGs. Figure 1 [6] illustrates the detailed operation of Media Gateways and the mapping between IP and traditional telephony networks.

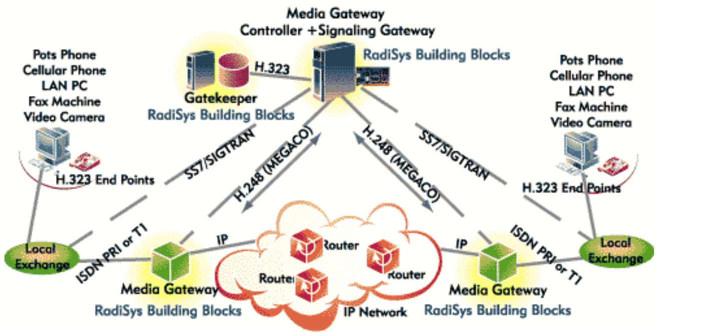


Figure 1: Media gateway, solution for the next generation telephony network [6].

MGC or "softswitch" is 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. Hence, next-generation networks must economically support both existing voice services such as CLASS 4/5 features and evolving applications such as integrated access, virtual private network, Internet call waiting, click to dial, unified messaging, and enhanced roaming.

MG implements the media mapping and transformation function identified in the decomposed gateway architecture proposed by ETSI TIPHON [9] and later adopted by IETF [4]. MG is the gateway that allows communication between two different networks, e.g., IP and public switched telephone network.

MGs can communicate via a real-time transport protocol (RTP) [8] that provides end-to-end transport functions suitable for applications transmitting real-time data such as interactive audio and video. RTP service is further augmented by real-time control protocol (RTCP) [8] to allow monitoring the data delivery.

1.2. Road map

In this paper, we describe OPNET implementation and simulation of Megaco/H.248 protocol [10]. Our goal is to gain experience in network simulation using OPNET and to explore current technologies in high-performance networks. We also employ software project management approach in completing the simulation project.

We also implement the RTP protocol in order to send and receive data between two MGs. In our simulation scenario, we employ a simple local area network with one MGC and two MGs. The system architecture is shown in Figure 2.

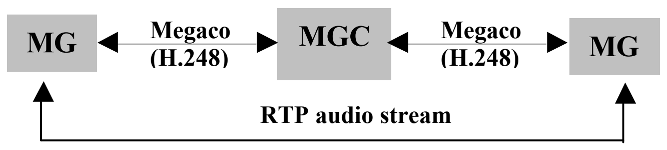


Figure 2: Simulated Megaco system architecture.

In Section 2, we provide details of Megaco functional requirements. Based on these requirements, in Section 3, we describe the architecture of MGC and MG. Next, in Section 4, we describe the basic call flow scenarios used in OPNET simulation. In Section 5, we describe the OPNET implementation. Simulation results are presented in Section 6. We discuss difficulties faced during and after the OPNET implementation and propose possible solutions in Section 7. Conclusions and discussion of possible future directions is given in Section 8.

2. Implementation requirements

We followed software engineering practice to clarify the requirements before OPNET implementation. We highlight here only the basic requirements. The complete specifications can be found in [10].

Basic requirements of this project are:

Simulation

- OPNET Modeler 8.0.C (Build 1252).
- OPNET run on a host named payette.ensc.sfu.ca.

Network

- Bus LAN (802.3).

Transport

- ALF (Application Layer Framing) over UDP between MGC and MG.
- RTP (Real-time Transport Protocol) over UDP between two MGs.

Message

- Augmented Backus-Naur Form (ABNF) text encoding of Megaco protocol.
- Each message includes:
  - ✓ Protocol name
  - ✓ Version number
  - ✓ IP address of sender
  - ✓ UDP port number of sender
  - ✓ One transaction.

Transaction

- Each transaction includes:
  - ✓ TransactionID
  - ✓ One context.
- Each transaction is 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 has the same TransactionID.
- TransactionID requirement:
  - ✓ TransactionID is assigned by sender
  - ✓ TransactionID is unique within the scope of sender
  - ✓ The value of TransactionID is from 1 to 99999, inclusively.

Context

- Each context has:
  - ✓ 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 "$".

Command

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

MGC

- There is only one MGC.
- The interface between MGC and MG is defined in System Interface Control Document (SICD) [10].

MG

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

RTP

- Two types of RTP packets are used:
  - ✓ RTP packet
  - ✓ RTCP packet.
- RTP packet has:
  - ✓ Fixed length header
  - ✓ RTP payload.
- RTCP packet has:
  - ✓ Fixed length header
  - ✓ RTCP packet type.

In Section 3, we discuss the architecture of MGC and MG in detail.

3. Architecture

MGC and MG are the two main components of our implementation. The architecture of these two components is based on Software Functional Specification (SWFS) of MGC, MG, and RTP [10].

3.1. MGC architecture

MGC, shown in Figure 3, has three main sub-components: Message Analyzer (MA), Message Sender (MS), and Message

Receiver (MR). The main control intelligence of MGC is located in the message analyzer, which implements all five commands discussed in Section 2. Upon receiving message from MG, MGC first extracts the message and interprets the commands. Then, based on the received commands, it takes actions accordingly. The communications between the MGC sub-components are local function calls.

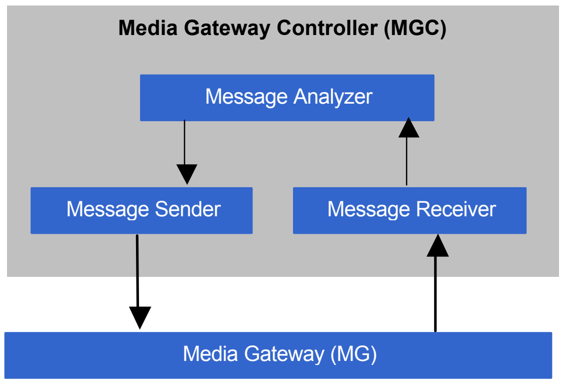


Figure 3: The architecture of MGC.

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.

3.2. MG Architecture

MG [2], shown in Figure 4, has four main sub-components: MG Processor, RTP, Audio Transducer, and User Interface.

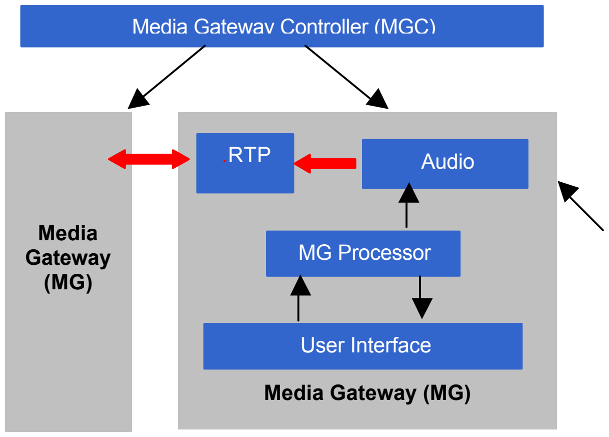


Figure 4: The architecture of MG.

The responsibilities of MG Processor are:

- 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 and signals.

Audio Transducer is responsible for generating the data traffic. RTP encodes the data traffic according to the encoding schemes specified in [7] and sends the RTP payloads to other MG. User interface provides the communication between user and the system.

4. Call flow scenarios

We implement and simulate three call flow scenarios in a Megaco network. The scenarios are: Successful MG Registration Procedure, Successful Call Setup Procedure, and Successful Call Release Procedure. We describe here each call flow as outlined in software specifications [10]. Figures 5, 6, and 7 show the three call flows. These figures illustrate Megaco commands and main parameters with their values.

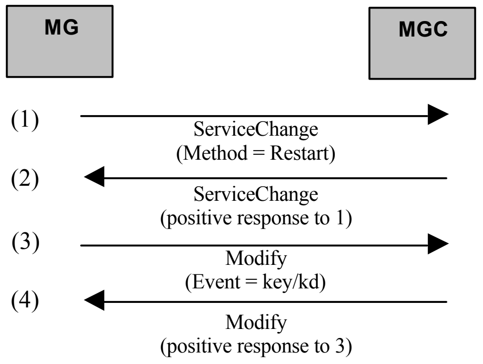
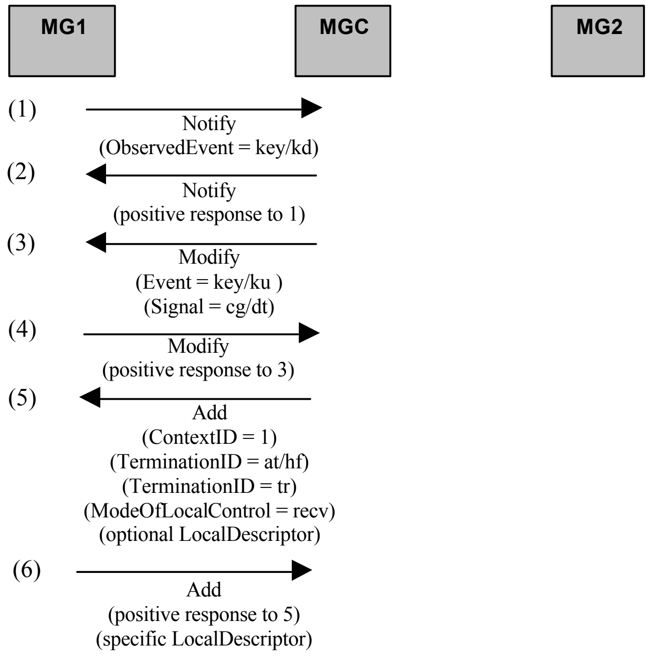


Figure 5: Successful MG registration procedure.



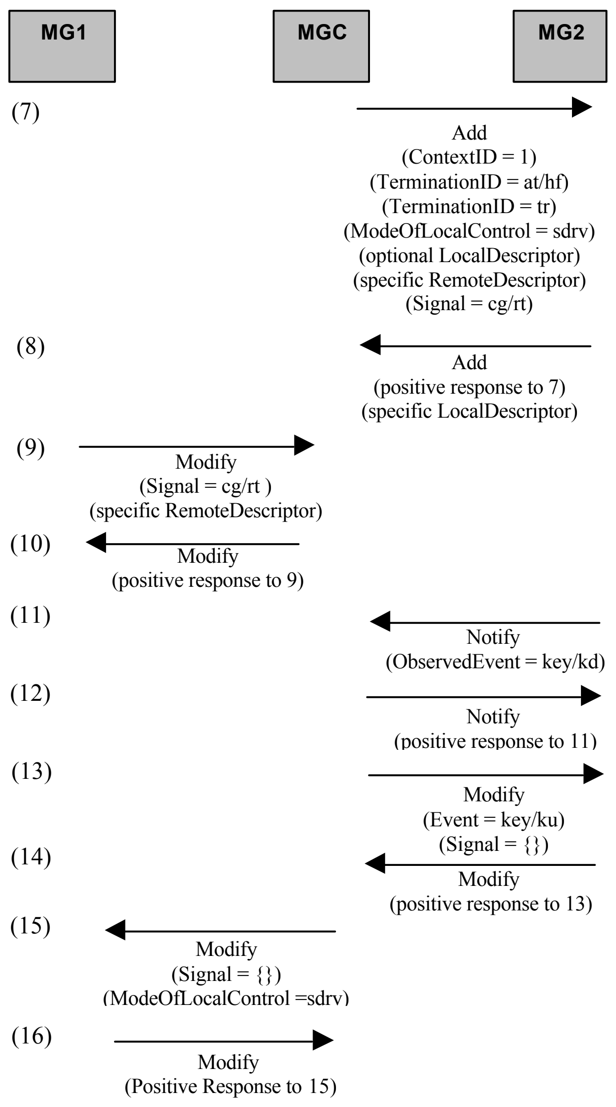
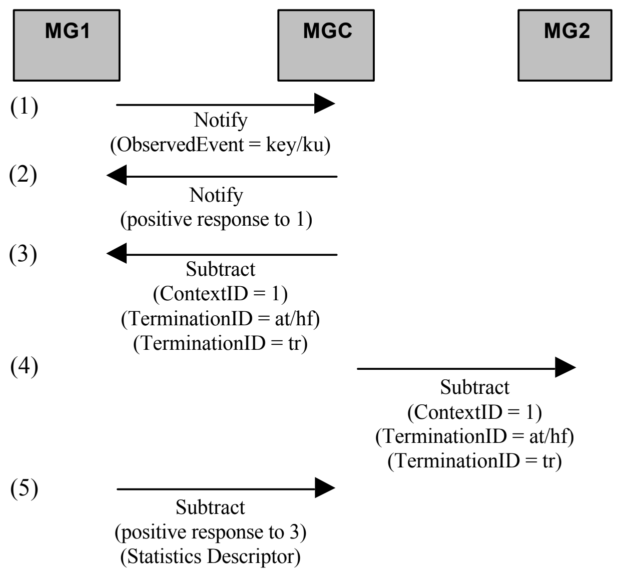


Figure 6: Successful call setup procedure.



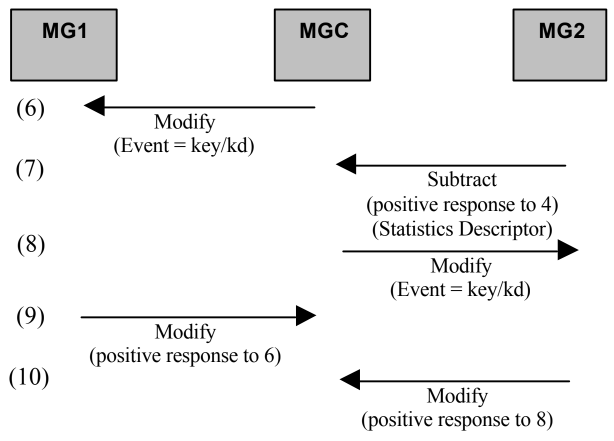


Figure 7: Successful call release procedure.

5. Implementation in OPNET

We describe here details of OPNET implementation.

5.1. Network model and node models

We build a simple network model to simulate Megaco [3] protocol in the OPNET simulation environment based on the software design specification of MGC, MG, and RTP [10]. Figure 8 illustrates the OPNET network model and simulation scenario. There are three nodes connected to a local hub by direct point-to-point duplex links. This is a master-slave architecture where top node is the master node (MGC) and two other nodes are slaves (MGs).

The protocol used between MGC and MG is Megaco/H.248 as specified in the standards specifications [1, 3, 5]. RTP protocol [8] is used for the communication between two MGs.

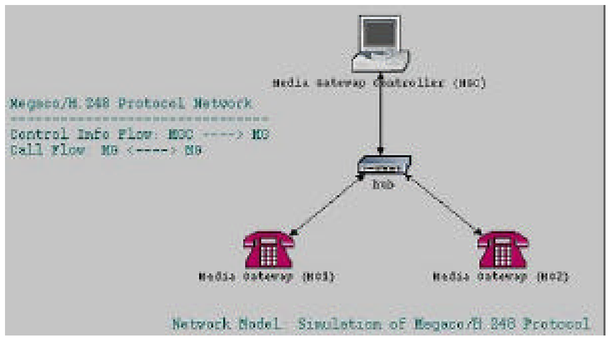

Media gateway controller (MGC)
Megaco/H.248 Protocol Network
Control Info Flow: MGC ----> MG
Call Flow MG <----> MG
hub
Media gateway (MG1)
Media gateway (MG2)
Network Model Simulation of Megaco/H.248 Protocol

Figure 8: Network model.

Figures 9 and 10 illustrate the node models of MGC and MG, respectively, in OPNET simulation environment. Each model has only one layer (application), one transmitter, and one receiver. The application layers of MGC and MG node models contain MGC and MG process models, respectively.

The hub node model is shown in Figure 11. It has three transmitters, 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 MGC and for each MG. For example, MGC_xmt and MGC_recv are connected to MGC transmitter and receiver, respectively.

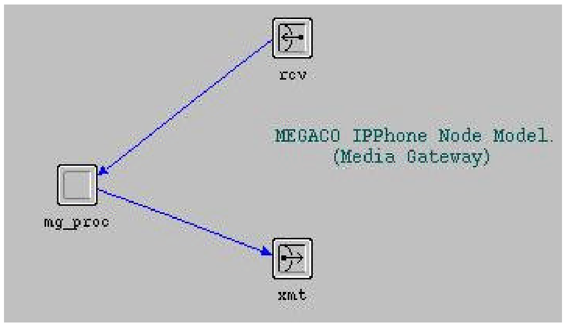


Figure 9: MGC node model.

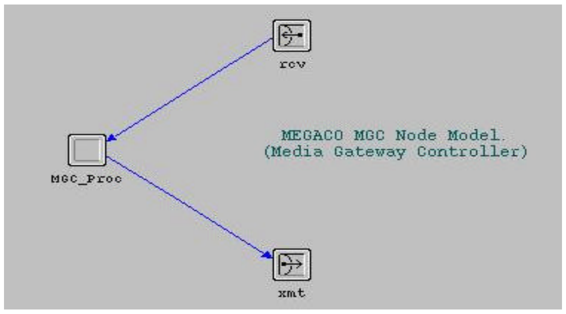


Figure 10: MG node model.

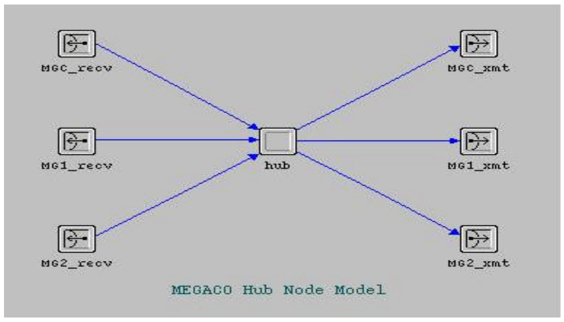


Figure 11: Hub node model.

5.2. MGC process model

The MGC process model, shown in Figure 12, is a finite state machine (FSM) that has three states: *init*, *idle*, and *process*. *Init* state initializes the MGC. MGC FSM then enters the *idle* state where it awaits the arrival of Megaco packet. Upon the arrival of a Megaco packet, the FSM moves to *process* state where all the MGC control intelligence is located. Here, the packets are unpacked and parsed. Afterwards, the *process* state interprets the command type and takes necessary steps depending on the commands.

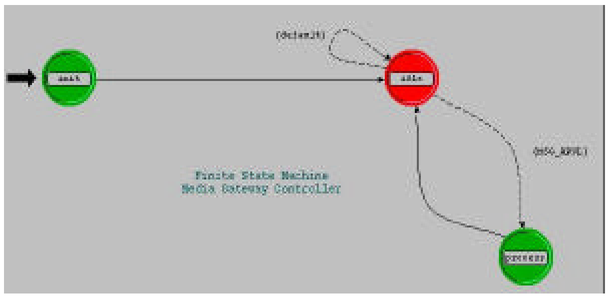


Figure 12: MGC process model.

5.3. MG process model

The MG process model, shown in Figure 13, is more complex than the MGC model. Its six states are *init*, *idle*, *MG_Proc*, *RTP_Proc*, *MG_Send,* and *MG_Recv*:

- *init* state initializes the MG. After the initialization, MG sends a ServiceChange request to MGC in order to register itself.
- *idle* is the next state, where MG waits for the acknowledgement of the registration request from MGC. After receiving the registration acknowledgement, MG awaits further commands from MGC and listens for the occurrence of key up or key down event. It also awaits RTP packets after the connection between two MGs is established.
- *MG_Proc* is responsible for interpreting the Megaco commands received from MGC and for acting accordingly. When MG detects a key down event, it notifies MGC about the event and sets up the connection according to the instructions from MGC. When the connection is established successfully, control is passed to *RTP_Proc* that is responsible for generating data traffic between two MGs.
- *RTP_Proc* process builds the RTP packets that carry the RTP payload according to the standard RTP packet format [8].
- *MG_Send* is responsible for sending Megaco and RTP packets to the output stream using the underlying transport mechanism. While in *idle* state, *MG_Recv* extracts a packet from the input stream when there is a packet arrival interrupt.
- *MG_Recv* retrieves the packet format and moves to either *MG_Proc* or *MG_Recv* state, depending on the packet sender.

When *MG_Proc* detects a key up event, a Notify command is sent to MGC in order to terminate the connection between two MGs. Then, MG receives Subtract command from MGC and, finally, *MG_Proc* tears down the connection.

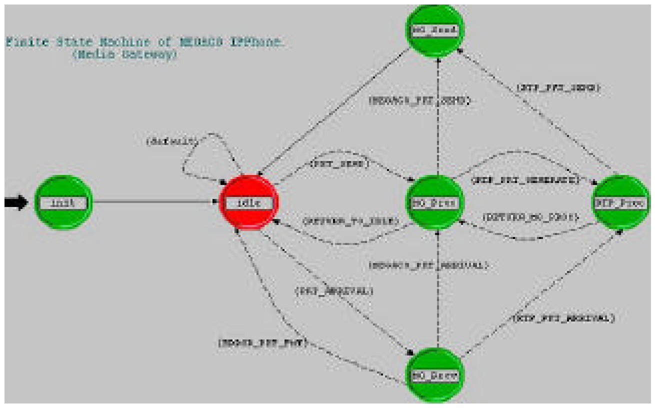


Figure 13: MG process model.

6. Simulation results

In this Section we present OPNET simulation results. These results are explained in Section 4. Simulation results of Successful Registration Procedures for two MGs, Successful Call Setup Procedure between two MGs, and Successful Call Release Procedure between two MGs are listed:

- Messages in Successful MG1 (IP address: 172.16.0.2) Registration Procedure: [1], [2], [3], and [7].
- Messages in Successful MG2 (IP address: 172.16.0.3) Registration Procedure: [4], [5], [6], and [8].
- Messages in Successful Call Setup Procedure: from [9] to [24].
- Messages in Successful Call Release Procedure: from [25] to [34].

We also show MG1 and MG2 statistics between messages [31] and [32], and messages [29] and [30], respectively.

```
|-----------------------------------------------------|
| [ 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
|----------------------------------------------------|
```

7. Problem summary and suggested solutions

7.1. MGC process model

Initially, we planned to implement Megaco protocol based on the layered architecture. The goal was to implement the application layer and to interface it with the standard OPNET UDP layer using Interface Control Information (ICI) package. It proved to be difficult to implement the protocol using all layers. The original network model is shown in Figure 14. In this network model, we used standard Ethernet hub and link models.

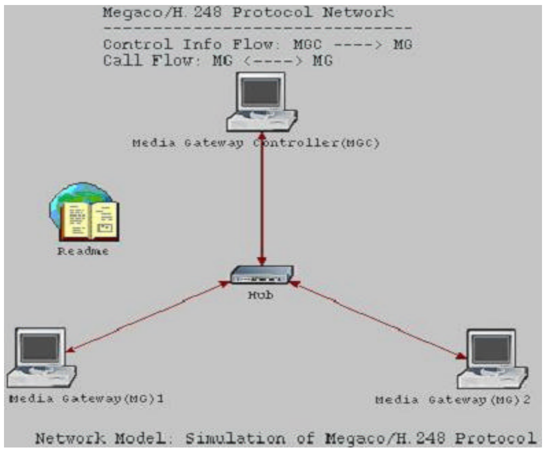

Megaco/H.248 Protocol Network
Control Info Flow: MGC ----> MG
Call Flow: MG <----> MG
Media Gateway Controller(MGC)
Readme
Hub
Media Gateway(MG)1
Media Gateway(MG)2
Network Model: Simulation of Megaco/H.248 Protocol

Figure 14: Initial network model.

Figure 15 describes the initial OPNET node model. It consists of six layers. The application layer contains all the functions that we implemented in OPNET: MGC, MG, and RTP. RTP is not a separate component. It is, instead, integrated within the MG. The application layer needs to communicate with lower layer transport mechanism in order to send and receive packets in an IP network. In our initial network model, we employed the UDP protocol as the underlying transport mechanism. However, we could not simulate our network using this node model due to OPNET difficulties in interfacing between layers. The problem could also be in the lack of understanding the simulation attributes of different layers. An OPNET tutorial involving communications between two layers may be helpful. We decided to modify the architecture of the node model and adopt a simple model with only application layer (as mentioned in Section 5.1). We made small changes in the process model in order to incorporate the new network model. Although we changed the initial node model architecture, we successfully simulated the Megaco/H.248 protocol as planned.

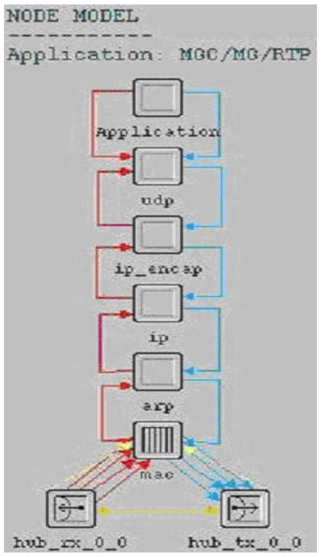


Figure 15: Initial node model.

7.2. OPNET interrupts

Interrupts play an important role in the design of FSMs in OPNET. They are essential in order to check the transition condition from unforced states. In the FSM of MG, we used self-interrupt to transition from *idle* to *MG_Proc* state in order to send ServiceChange and Notify commands to MGC because they do not depend on packet arrival. We set the <intrpt interval> attribute of the MG process model to 20, hence scheduling an interrupt for the root process of the processor module after every 20 seconds.

7.3. Transition condition

This condition requires special care to ensure that more than one transition condition does not become true in a specific state simultaneously.

8. Conclusions and future work

In this paper, we described OPNET implementation of Megaco/H.248. We have implemented five Megaco/H.248 commands and simulated three basic call flows in Megaco: Successful MG Registration Procedure, Successful Call Setup Procedure, and Successful Call Release Procedure. We described the OPNET implementation details and presented simulation results.

Future improvements of the Megaco/H.248 OPNET model could include implementation of:

- Remaining three Megaco/H.248 commands
- Full featured RTP
- Multiple call scenarios
- Layered architecture.

References

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

[2] M. Brown and 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).

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

[4] N. Greene, M. Ramalho, and B. Rosen, "Media Gateway Control Protocol Architecture and Requirements," RFC 2805, April 1999: http://www.ietf.org/rfc/rfc2805.txt (accessed in August 2002).

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

[6] "Radisys Building Blocks for Media Gateway Solutions," http://www.radisys.com/files/media_gateway.swf (accessed in August 2002).

[7] H. Schulzrinne, "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 August 2002).

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

[9] Telecommunications and Internet Protocol Harmonization over Networks (TIPHON), http://www.etsi.org/tiphon (accessed in August 2002).

[10] S. Wu, M. Riyadh, and M. Mannan, "OPNET Implementation of Megaco/H.248," http://www.sfu.ca/~vswu/courses/CMPT885/project.htm (accessed in August 2002).