A Multi-Signaling Protocol Architecture for Voice over IP Terminal

Anoop Kumar K. [1] and Tanu Malhotra[2]

Wireline Systems Group, R&D Centre,
STMicroelectronics AP Pte. Ltd., 20, Science Park Road,
#01-28 to 30 Teletech Park, Singapore Science Park 2, Singapore – 117674.
{anoop.kumar, tanu.malhotra}@st.com

Abstract- **This paper describes a novel multi-signaling protocol architecture of a VoIP endpoint in an IP packet network. The endpoint is a single channel IP phone or gateway, but can handle H.323, SIP or MGCP calls simultaneously to/from different endpoints. The other endpoints could be using any of the signaling protocols. The endpoint is registered to a Media Gateway Controller, SIP server and a H.323 gatekeeper, possibly with the same number. The details of the design including the state machines of the different call control modules are described. The advantage of such architecture is reduced delay in call setup between two endpoints that otherwise use different signaling protocols. The architecture has been implemented and verified on one of the company's IP Phone chip, which has an ARM7 and a DSP core. An example of a simple handling of a second call from a different protocol endpoint, when the first call is in progress, has also been given.**

Index terms— **VoIP, Signaling Protocols, H.323, SIP, MGCP, Endpoint, Media Gateway, Gatekeeper, SIP server, Media Gateway Controller.**

1 INTRODUCTION

The explosive growth of the Internet and applications based on the Internet Protocol saw a rapid change in the communication industry. Packet-based-voice has evolved over the years to enable a single network to deliver a new generation of business and consumer voice and data services. New networks are emerging based on packet technology that supports a converged network that carries both voice and data. The total amount of packet-based network traffic has exceeded the traditional circuit-switched network traffic. Due to the technological advancements, in voice over IP (VoIP) or IP telephony, it is clear that the voice traffic and services will be one of the major applications. The well-known benefits of VoIP are cost savings, open standards and multi-vendor interoperability and integrated voice and data services.

There are several groups that contributed to the standard VoIP protocols. In particular, four different signaling and call control protocols for VoIP are: 1) H.323, 2) Session Initiation Protocol (SIP), 3) Media Gateway Control protocol (MGCP) and 4) H.248 / MegaCo. To get workable VoIP solutions, a great deal of effort had to be put to determine which protocol was best suited for particular networks and applications. Although it is highly desirable that there should be only one voice protocol for each function in a packet network, in reality multiple VoIP protocols and architectures have already been deployed and many networks will continue to be built using different VoIP protocols.

As companies expand their networks, they are faced with choices about how to interconnect segments using differing VoIP protocols. These fall into one of three categories [6]: (i) Time Division Multiplexing (TDM) – the problem with this is that it introduces latency. (ii) Single Protocol Architecture – In this model, a company moves all its VoIP devices and services to a single protocol, simplifying the network as a whole. But this limits the potential connectivity to other networks that are using other VoIP signaling protocols. (iii) Protocol translation – Use of IP-based protocol translators to interconnect two or more VoIP protocol domains. But there is no standard for protocol translation and so not all VoIP translators are exactly same.

Companies choose vendors who are not only committed to supporting open standards but also those who are actively developing voice strategies that consider interoperability with all the VoIP protocols. Customers need products that support multiple protocols and vendors must provide solutions that work in both single-protocol and multi-protocol environments [6]. It is not clear how the VoIP signaling architectures will eventually evolve. But it is clear that these different protocols will need to coexist for sometime [2], and the evolution will be determined by market forces.

In this paper, an endpoint management architecture is proposed which supports multiple signaling protocols for Voice over IP. The endpoint is registered to a Media Gateway Controller (MGC), a SIP server and a H.323 gatekeeper at the same time. The proposed endpoint can handle calls, simultaneously, from other endpoints that support different protocol. For example, if the MG endpoint is in a call with another Media gateway (MG), through a MGC, and a SIP user agent (UA) calls in through a SIP server, then the calling SIP UA would hear a busy tone. The advantage of such architecture is reduced delay in call setup between two endpoints that otherwise use different signaling protocols. For example, if the H.323 endpoint call a SIP endpoint, then an InterWorking Unit (IWU) is used to implement the bridge between the H.323 and SIP call setup signaling. This IWU needs to also have an implementation of the ASN.1 encoder/decoder as well as ABNF encoder / decoder. For the SIP messages it has to decode ABNF messages and encode it to ASN.1 messages to send it to the H.323 terminal. Similarly, for the H.323 messages it has to decode ASN.1messages and encode it to

1. IEEE SM'03. 2. Corresponding Author

ABNF messages to send it to the SIP terminal. Apart from this processing delay, there could be network delay depending on the physical location of the IWU. The call setup itself depends on the availability of the IWU. Also, even if one among the MGC, SIP server or GK, to which the endpoint is registered, is non-functional, the endpoint can handle calls using the other protocol.

The paper is organized as follows: section 2 gives an overview of the three single protocols that are used in the present VoIP systems, namely H.323, SIP and MGCP. Section 3 describes, in detail, the proposed architecture of the multiple protocol endpoint system. Section 4 describes the implementation and verification of such an endpoint and section 5 contains the concluding remarks and scope for future work.

2 OVERVIEW OF VOIP SIGNALING PROTOCOLS

In this section a brief description of the H.323, SIP and MGCP protocols are given, defining their main components and functionalities. For details of the call flow etc., refer to [3] for H.323, [4] for SIP, [5] for MGCP and [1, 8, 10] for consolidated information on VoIP.

H.323: The H.323 is an ITU-T standard [3] for multimedia communications over local area networks that do not guarantee Quality of Service (QoS). It addresses call control, multimedia management, bandwidth management and interfaces between LANs and other network. The entities in the H.323 topology are the endpoints, also called user terminals, gateways (GW), multipoint control units (MCU) and gatekeeper (GK). Gateways perform the translation of signaling and media exchange between H.323 and PSTN endpoint. MCUs, though listed separately, are in practice part of a gatekeeper or a high-speed computer that acts as a terminal serving one or more users [8]. Gatekeepers are responsible for call authorization, address resolution and bandwidth management. The H.323 standard defines procedures for user registration with a gatekeeper, call control and logical channel capabilities negotiation between two or more parties. The Registration, Authentication/Admission and Status (RAS) and the call control are as per the standard H.225. The logical channel capabilities negotiation is done as per the H.245 standard.

Usually, in real-time communication between two endpoints, the two endpoints are H.323 terminals. But, in general, the communication could be in a dissimilar domain – PSTN, SIP or MGCP, using a gateway. The H.323 call model is between two H.323 endpoint terminals, a terminal and a gatekeeper, or a terminal and a gateway. A VoIP service architecture for integrated communication between IP-based telephony with the circuit-switched intelligent network has been proposed in [7].

A brief description of the other protocols used by the H.323 is given: (i) The RAS is a transaction-oriented protocol between an H.323 endpoint and a gatekeeper. The endpoint uses RAS procedures to discover a gatekeeper, register or unregister with a gatekeeper, requesting call admission and bandwidth allocation, and call termination. A gatekeeper can enquire the status of an endpoint using RAS. (ii) The H.225, which uses a subset of the ITU-T Q.931, is used for the call setup and teardown between two H.323 terminals. (iii) The H.245 is used for connection control, allowing two endpoints to negotiate media processing capabilities of the terminal, such as audio/video codec, for each media channel between them. It also determines master-slave relationships of endpoints, and open and close logical channels between two endpoints.

SIP: The Session Initiation Protocol [4] is a text-based signaling protocol. SIP is a client server protocol transported over either TCP or UDP, but common implementations use SIP over UDP for simplicity and speed [8]. SIP is a simpler protocol than H.323. The SIP consists of user agents (UA) and one or more servers. The initiator of a SIP request is called a SIP client and the responding entity is called a SIP server. First, a signaling connection between the calling and the called endpoint is opened. SIP call control uses Session Description Protocol (SDP) [12] to describe the details of the call. The body of the SIP message contains a description of the session, such as media type, codec type, packet size etc., in the SDP format. SIP uses a Uniform Resource Identifier (URI) to identify a logical destination, not an IP address. SIP can notify users of events and also allows easy addition of new services, like sending instant text messages [9].

Some of the important SIP functional entities are: (i) User Agent: This entity acts as a SIP client when a SIP request is initiated and as a SIP server when a SIP request is received. (ii) SIP Proxy: This entity acts as a SIP client and a SIP server in making SIP requests on behalf of other SIP clients. (iii) Registrar: This entity is a SIP server that receives, authenticates and accepts REGISTER requests from SIP clients. It may be together with a SIP proxy. (iv) Location Server: This entity stores user information in a database and helps determine where to send a request. (v) Redirect Server: It just responds to a SIP request with an address. The requester then contacts the address directly.

Though the SIP protocol has shown some promise, in a shared network there are some security concerns like preserving confidentiality and integrity of SIP requests, ensuring privacy of participants in a session, etc.

MGCP: MGCP is a master/slave protocol [5, 10]. The call processing functions are separated from the gateway functions in an entity called a “call agent”. The call agent controls the media gateway. The call agent need not be located physically near the gateway. The call agent could also control many gateways. This architecture simplifies the functionality of the gateways. MGCP also uses the session description protocol (SDP) to describe the media aspect and various parameters to enable establishment of connections between endpoints. MGCP does not use the multimedia features of SDP such as the multiparty multimedia conferences, which is supported in H.323 and SIP [10]. Although the MGCP is designed as a distributed system protocol, it gives the user the appearance as a single VoIP system. The MG part of the MGCP protocol is stateless, in the sense that the sequence of transactions with the MGC can be performed without any memory of the previous

transactions. But MGCP requires the MGC part to keep the call state.

For all the three VoIP protocols mentioned above, RTP is the de facto standard [11] mechanism to exchange media payload. RTP does not guarantee QoS and does not address resource reservation along the path of the connection.

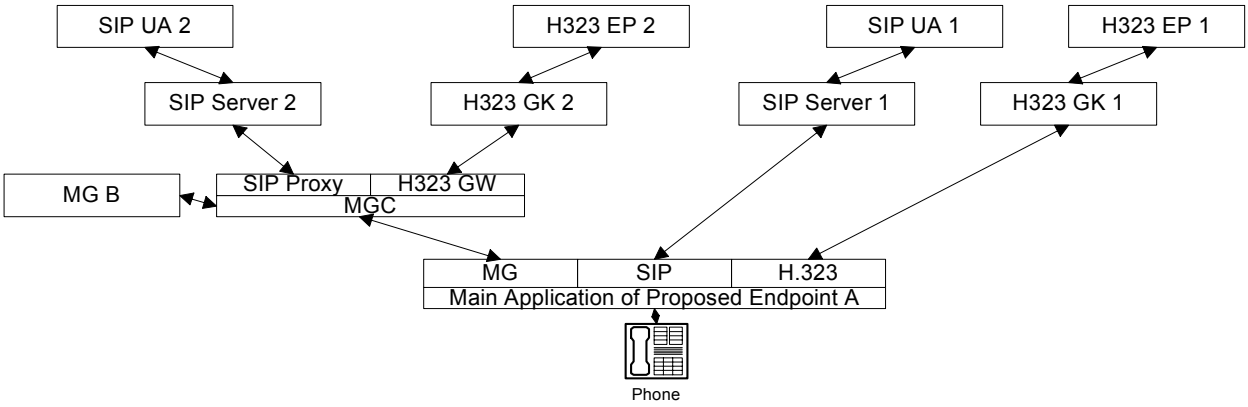

Figure 1 Typical Network Topology

3 MULTI-PROTOCOL SIGNALING ARCHITECTURE

The proposed endpoint supports, in this case three signaling protocols namely H.323, SIP and MGCP. A straightforward design would be to support three channels, each supporting a particular signaling protocol, or a system supporting just a single protocol for example as given in [13] for H.323. But in the method proposed here, there is effectively only a single channel viz. a single telephone. A typical network topology for such a case is shown in Fig. 1. In this figure, Endpoint A (EP A) is the proposed endpoint, the details of which would be explained in this section. The topology consists of a H.323 Gatekeeper (GK1), a SIP Server (SIPSERV 1) and a Media Gateway Controller (MGC). The MGC also comprises of a SIP proxy and a H.323 Gateway, which could be connected to another SIP Server (SIPSERV2) and another H.323 Gatekeeper (GK2) respectively. Endpoint A is registered to the MGC, SIPSERV 1, and Gatekeeper (GK1). There could be different call scenarios. For e.g. An MG (MG B) could call in to EP A. When this call is in progress, if a SIP UA calls in through the SIP server 1, or a H.323 EP calls through the GK1, then these calling-in EPs should hear a busy tone. In other words, EP A can communicate using any of the VoIP signaling protocols, but when any call is in progress, any other incoming call should hear a busy tone.

Figure 2 shows the overall system software architecture. The main controller is the Task Manager (TM) which is responsible to startup the relevant modules in the application. As is evident from the figure, it initializes and starts the Ethernet drivers, the software bridge and also launches each of the Sub Task Managers (SubTMs) viz., Phone Manager Sub Task Manager (PhoneSubTM), Application Control Sub Task Manager (ACSubTM) and Telephony Control Sub Task Manager (TCSubTM). Each of these is a separate task and the inter-task communication is accomplished using First In First Out (FIFO) message queues. TM supervises all these tasks at the runtime and checks if all the tasks are alive. It is the responsibility of each of the SubTMs to send an acknowledgement back to the TM to confirm the successful launch of each of their sub modules.

The ACSubTM launches the Display Manager, Keypad Manager and the "User Interface Control" (UI Control) threads. The H.323, SIP and MGCP Call control modules are separate threads that are initiated by the TCSubTM. The TCSubTM also starts another thread called the Voice Manager (VM). The UI Control thread has a message queue that keeps checking for messages from the Call Control (CC) module and the Message Decoder (MsgDecoder) module and performs appropriate actions. The role of Keypad Manager is to manage keypad events (key pressed/released), manage DTMF events detected by Phone Manager and request Phone Manager to detect DTMF, if enabled. The CC modules interact with the respective signaling protocols (H.323, SIP, MGCP) by using the Application Program Interface (APIs) provided by the stacks. Also, any messages received by the H.323/SIP/MGCP stack are informed to the respective Call Control modules using the Call back functions. The function pointer is registered to the corresponding stacks right at the time of initializing and starting its instance. This call back function is implemented in the Call control module of the respective stacks.

The role of PhoneSubTM is to launch all the tasks of Phone Manager Module viz., MsgDecoder and Codec Manager, and tests the interfaces to the DSP. DSP hosts the DTMF detection / tone generation, speech codecs and other voice functions. The MsgDecoder also has an interface to the UI Control and the Call Control modules. The purpose of Msg Decoder is to dispatch messages either to DSP or to Codec Manager. The role of Codec Manager is to pass commands to Codec in order to perform specific features like ring the phone, start voice processing, play DTMF tones, tune the gain (voice volume) and to catch Codec events (hook movements). MsgDecoder also dispatches messages coming from DSP to the

corresponding software modules. For e.g., on receiving the DATA message from the DSP for the digitized voice, it sends the payload to the VoiceTx module, which is a task in the VM, to be send to the remote side using the RTP APIs.

The Voice Manager initiates three tasks, namely the jitter buffer control, Voice Transmit (VoiceTx) and Voice Receive (VoiceRx). The jitter buffer algorithm puts the received RTP packets in an orderly fashion, based on the time stamp and the sequence number, before passing them to the DSP for decoding. The role of VoiceRx task is to wake up the RTP stack every 10ms and hence schedule the reception of voice packets (this will enable the system to handle voice packets up to the rate of one RTP packet every 10ms). The VoiceTx module has to wake up the RTP stack each time a voice packet is received from DSP and hence schedule the transmission of voice packets.

We limit our discussion to the Call Control modules of H.323/SIP/MGCP and the UI control module, as these are the modules that need design changes in the multi-protocol architecture case. The other modules are similar to existing single protocol cases. Figure 3 shows the state diagram of the UI control. The important states in this State Machine (SM) are Standby, WaitOnHook, Ringing, Busy Tone, Dialing and the Call state.

In the case of an outgoing call, we assume there are buttons to select an MGCP, SIP or H.323 call. The user selects one of these and this helps the UI control to know to which call control unit it needs to interact with. When the user goes off-hook for the outgoing call, the UI control first checks if there is a call available, which is checked against the maximum number of calls available (1 in our case). If there is a call available, the UI Control sends the M_OFF_HOOK message to the Call Control module where this message is handled appropriately according to the different VoIP Stack functionality. The UI control then checks if the call being made is a H.323/SIP call. If yes, the UI module immediately sends M_CB_DIALTONE to play the dial tone and to enable DTMF detection. The state changes from "standby" to "dialing" state. If the call being made is an MGCP call, the dial tone is not immediately played. The MGCP CC module, waiting for the messages in its message queue, on reception of the M_OFF_HOOK message from the UI module, sends OFF_HOOK event (EVENT_HD) to MGC and waits for any response from him. On receiving the EVENT_HD event from the MG, MGC should respond with playing of Dial tone signal (SIGNAL_DL). The callback function implemented in the MG CC module on detecting the SIGNAL_DL (ON) from the stack sends M_CB_DIALTONE to the UI module to be send to the message decoder and then to the DSP interface to play out the Dial tone and to enable DTMF detection. Now the state changes from "standby" to "dialing" state. At this stage, user dials the phone number of the calling party using the Keypad available. The DTMF tones are detected by the DSP and sent to the MsgDecoder and then to the UI Control module (if DTMF detection is enabled). Once the phone number is entered and the "dialing" is complete which in our case is represented by the number followed by "#" key, the state changes to the "Call" state and PLACE_CALL message is sent to the Call Control with the phone number to be called. If the user cancels dialing or takes unusually longer time to key in the phone number then the state times out and changes to "Busy Tone" state. In this state busy tone is played to the user and after a time out busy tone is stopped and the state changes to "WaitOnHook" state where it waits for the user to go on-hook. When the user goes on-hook the state machine resets to the "Standby" state. When the UI Control receives CallEvent(A,rem_ringing) message or

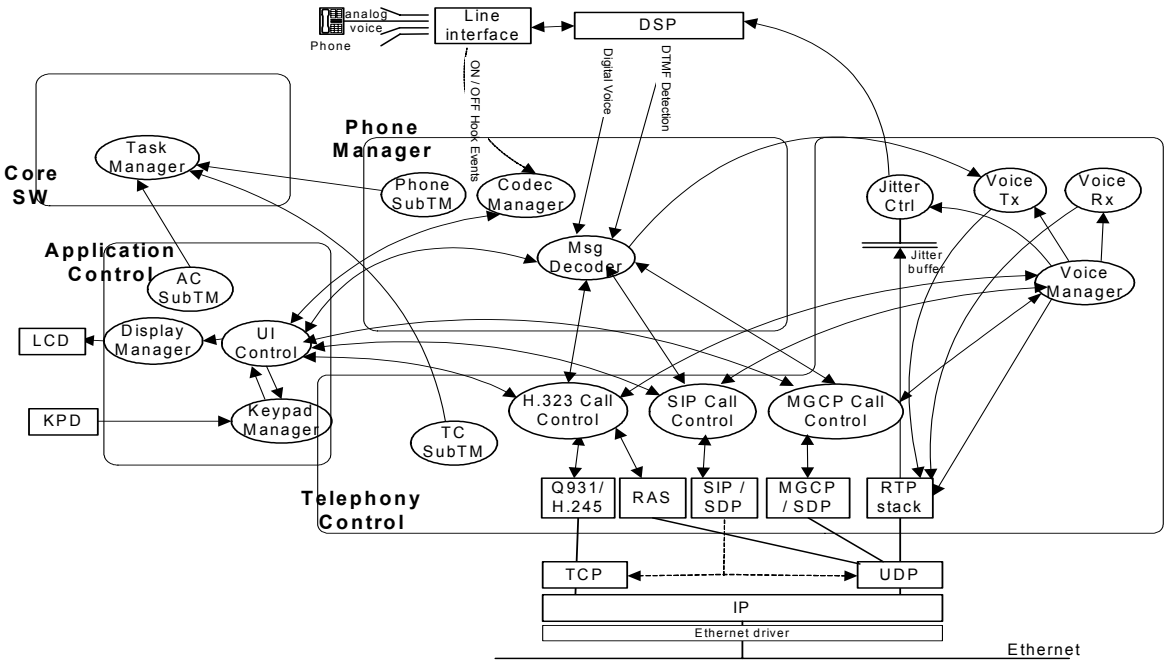

Figure 2 Overall Multi-Signaling Protocol System Architecture

<PROTOCOL>_CBKMSG_RINGING indicating that far end or remote side is ringing, ring back tone is played. On receiving the far end connect, the ring back tone is stopped and the state remains in the "Call" state and the user starts talking to the remote side user. If the remote phone is busy, then the control enters into the "Busy Tone" state where a busy tone is played and after a timeout, busy tone is stopped and the state is changed to "Wait On Hook" where it waits for the user to go on hook. This could also happen if the remote phone disconnects.

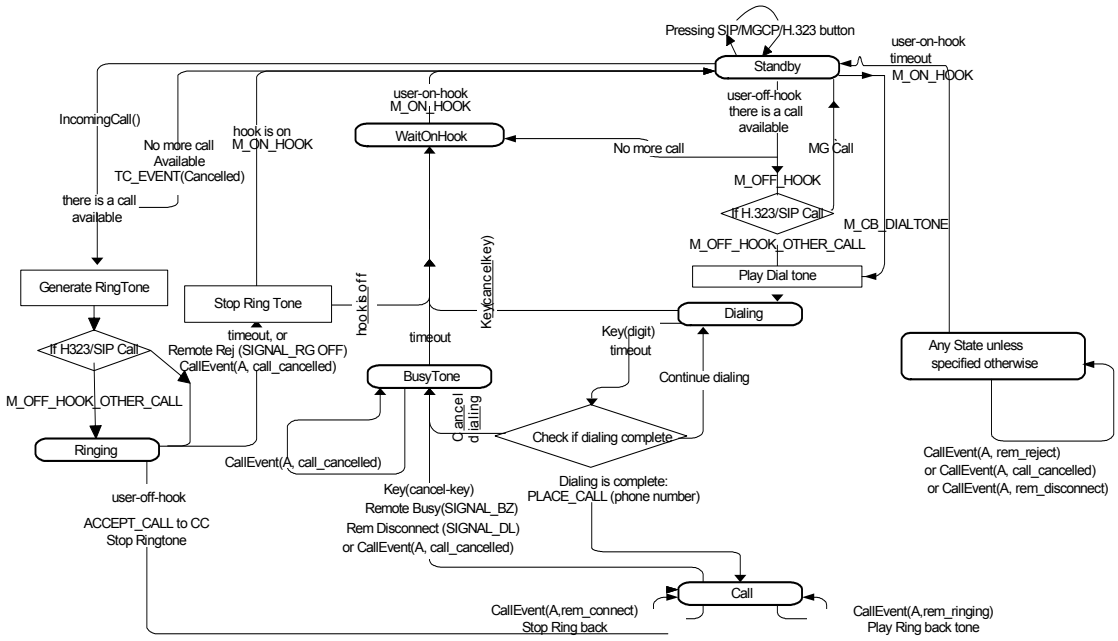

Figure 3 User Interface(UI) Control State Machine

In the case of an incoming call i.e. SETUP received in case of H.323 or INVITE received in case of SIP or Create Connection (CRCX) + Ringer Signal ON received in case of MGCP, the <PROTOCOL>_CBKMSG_INCOMING is sent to the corresponding Call control callback function and this Incoming Call message is forwarded to the UI Control. UI Control receives this message and if there is any free call available ring tone is generated, the number of free calls available is decremented to zero and the state is changed from "Standby" to "Ringing" state. Also, it displays the Caller Id information, if received, to the LCD display. If the incoming call is H.323/SIP call then M_OFF_HOOK_OTHER_CALL message is also sent to the MGCP CC module. In our present simple case, we assume that system can handle only one call at a time, with no supplementary services being implemented at present. If the number of free calls available is already zero then it remains in the "Standby" state and TC_EVENT (Cancelled) is send to the CC module. From the "Ringing" state, the state may also change to the "Standby" state after ringing is stopped due to a timeout (call is not accepted) or remote call rejection. Once the user goes off-hook ACCEPT_CALL message is sent to the H.323/SIP/MGCP CC modules. On receiving the M_CB_RINGING_OFF, the ringer is stopped and the media channels are opened for voice conversation and the state is changed to "call" state. If the remote hangs up first, then the control enters into the "Busy Tone" state where a busy tone is played and after a timeout busy tone is stopped and the state is changed to "Wait On Hook" where it waits for the user to go on hook. When user goes on-hook the state is changed to "Standby" state.

Figure 4 shows the state diagram for the Call Control modules of the SIP/H.323 protocol. The initial state is the "standby" state. When there is an incoming call, first the state changes to the "Wait Call Accept" and Incoming call message is sent to the UI Control message queue with the Caller Id information (if available). If TC_EVENT (Cancelled) message is received from the UI Control, then it means that there is no free call available and there is already a call going on, on some protocol stack. If all the three instances of the VoIP stack were free, then <Protocol>Call Control would receive the ACCEPT_CALL message. <Protocol> here refers to SIP/H.323. The appropriate <Protocol>AcceptCall API is called to perform the appropriate signaling to accept the call, viz. OK message in SIP or ALERTING/CONNECT message in H.323. The state is changed from "Wait Call Accept" to "Wait Media Start". In this state, if we receive the callback message for media start for RX or TX, the state remains the same. Only on reception of media start, for both RX and TX, the state is changed to "Wait Open RTP Channel". Media Start call back message for RX and TX contains parameters like the remote destination IP address, media type (audio/video), type of vocoders negotiated, packet size, etc for RX and TX side respectively. These parameters are passed to the voice manager to call the relevant APIs to start the RTP instances. On successful starting of the RTP instances, the Voice Manager sends an acknowledgement to the Call Control and the state is

changed to "Wait Open Voice Channel". On successful opening of the voice channels the state is changed to "Communication" state and the voice call is established. On receiving the remote disconnect or cancel call (local on hook) event is sent by the UI Control to the MGCP CC module. On reception of the M_OFF_HOOK message from the UI module, CC module sends an off hook event (EVENT_HD) to MGC and waits for any response from it. On receiving the

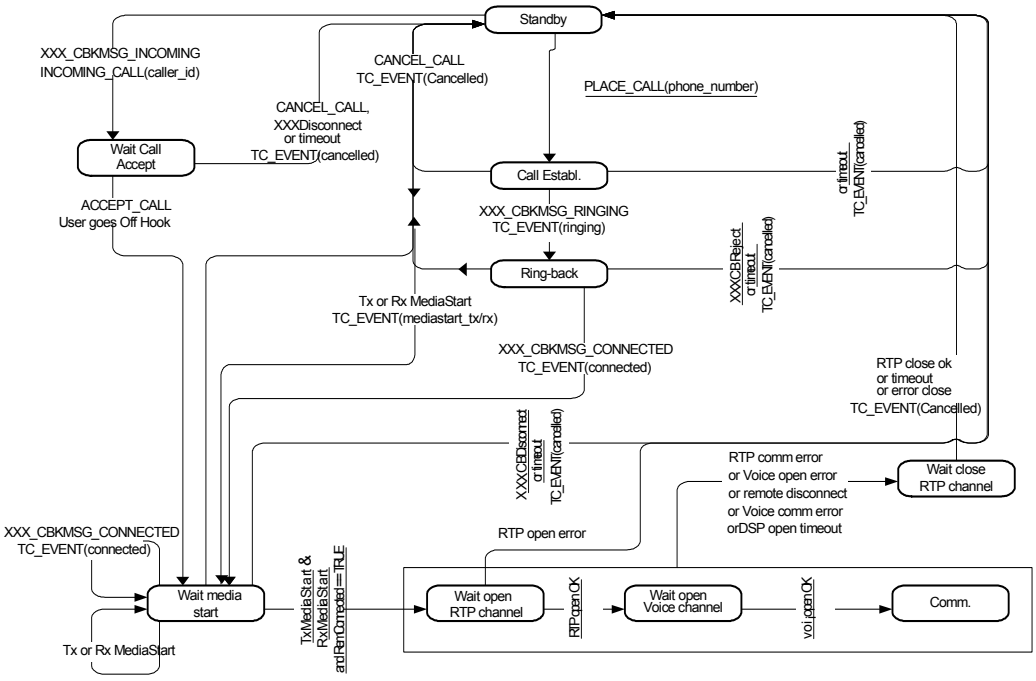

Figure 4 H.323/SIP Call Control State Machine Diagram

the state is changed to "standby" state. Also, in case of any errors in the opening of RTP channels or voice channels results in destruction of the RTP instances (if opened) and change of the state to "standby" state.

When the user makes an outgoing call (presses H.323/SIP button), the H.323/SIP CC module receives PLACE_CALL message with the phone number. The relevant APIs <PROTOCOL>CallPlace is called to make an outgoing call and the state is changed to "Call Establishment" state. Ring back tone is played when the far end starts ringing and on reception of the Media start call back messages for the TX and RX as explained above, ring back tone is stopped and the state changes, in sequence, to Communication state and the voice call is established.

Figure 5 shows the state diagram for the MGCP CC module. In the case of MGCP, when there is an incoming signal for the Create Connection (CRCX) and Ringer (RG) ON then Incoming Call message is sent to the UI module and the ring tone is played. The state is changed from "Standby" state to "Wait Call Accept" state. When the user goes off hook to accept the call, off hook event (EVENT_HD) is sent to the MGC and state is changed to "Wait Media start". On receiving the Media Start call back message for both TX and RX and on successful opening of RTP instances and the voice channels, the state is changed, in sequence, to "Communication" state.

When the user wishes to make an outgoing call (presses MGCP button), he goes off-hook and M_OFF_HOOK EVENT_HD event from the MG, MGC responds with playing of Dial tone signal (SIGNAL_DL) and enabling of DTMF detection. The callback function implemented in the MG CC module on detecting the SIGNAL_DL (ON) from the stack sends M_CB_DIALTONE to the UI module to be sent to the message decoder module and then to the DSP interface to play out the Dial tone and to enable DTMF detection. The CC module receives PLACE_CALL message along with the phone number when the dialing is complete and the state is changed to "Call establishment" state. Each digit in the phone number is sent to the MGC one by one as dtmf events for numbers pressed from 0 to 9 (EVENT_DTMF_0 – EVENT_DTMF_9). Once a match is found at the MGC side, for the called party, Create Connection (CRCX) is send to the MG endpoint. Once the far end starts ringing, ring-back tone signal (SIGNAL_RT) is sent, to play ring-back tone at the MG endpoint. On receiving the callback message for the media start for TX and RX, ring back tone is stopped and the RTP instances and the voice channels are opened. On successful opening of the same, the state is changed to "Communication" state and voice call is established.

After going off-hook to place the outgoing call or on accepting the incoming call, the user can go on-hook in any of the states. Thus, on receiving the M_ON_HOOK from UI Control, in any of such states, EVENT_HU is send to the MGC to reset the state machine at the MGC side. Also, after going off-hook if the SIGNAL_DL is received from the MGC it

means that the remote side has disconnected and the MGCPCBReject or MGCPCBDisconnect message is send to the UI Control and the state machine is reset to the "Standby" state.

If the MGCP CC receives the M_OFF_HOOK_OTHER_CALL in any of the CC module states, as in a situation mentioned earlier, then it simply means that there is an incoming call callback message received by the H.323/SIP CC module. This message is only handled in the "Standby" state and on receiving this message, MG Endpoint enters into the "Busy" state and sends an off-hook event (EVENT_HD) to make MGC believe that MG endpoint is busy and for any further calls destined to it, MGC should respond with a busy signal. When H.323/SIP module finishes with the call and goes on-hook, the M_ON_HOOK_OTHER_CALL is send to the MG CC module and it sends an on-hook event (EVENT_HU) to the MGC so that further calls destined to it are not responded with busy signal, instead a CRCX should be sent and the state changed back to "Standby" state.

Thus, we see that with the above design of a multi-protocol architecture, a call can be established, whether incoming or outgoing, using any one protocol (H.323/SIP/MGCP). If there is a second call coming-in from an endpoint using a different protocol, then the proposed endpoint will be able to handle it appropriately and the second calling-in endpoint hears a busy tone.

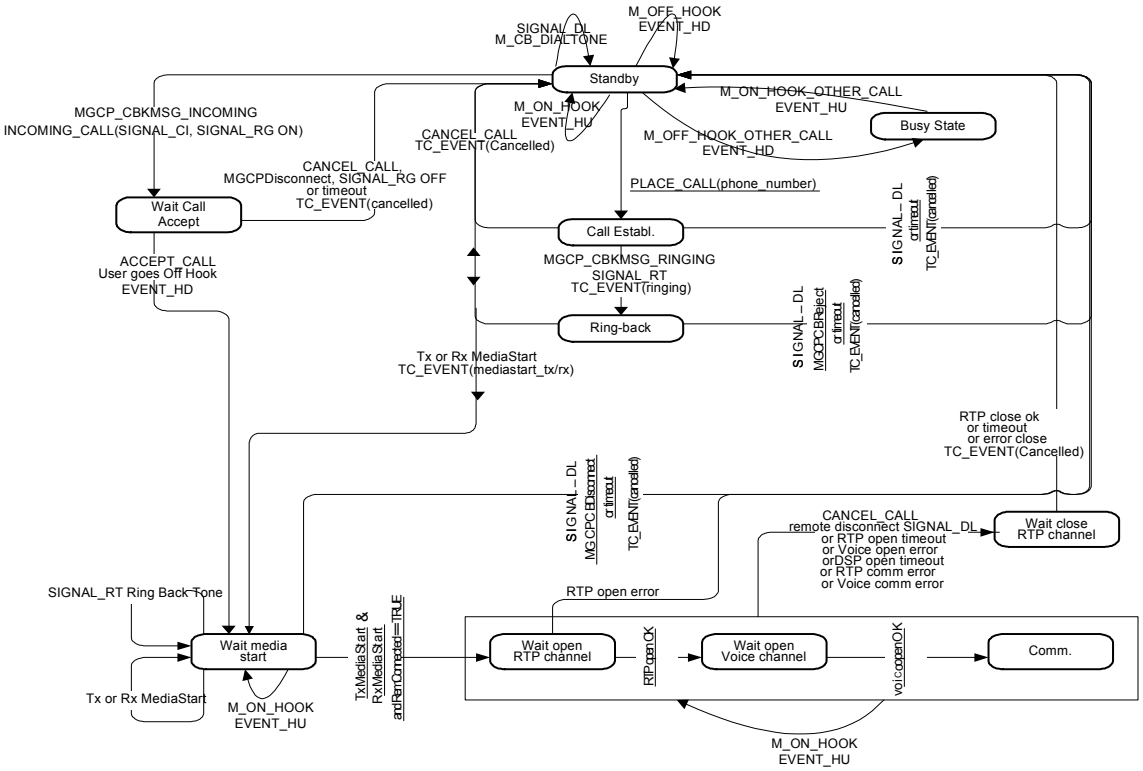

Figure 5 MGCP Call Control State Machine Diagram

4 IMPLEMENTATION DETAILS

The system consists of an ARM7TDMI core and a DSP core. The implementation of the H.323/SIP/MGCP stack, which is reentrant, is on ARM7 microcontroller. The speech vocoders are implemented on a DSP. The RTP/RTCP stack is also reentrant and supports multiple instances using the same task. If we have a H.323 single protocol implementation, the only additional modules needed for the multi-protocol implementation are the SIP, MGCP stacks, their CC modules and the SDP stack. All other modules are common and need minimal changes, which include the RTP/RTCP, NTP stack and all the DSP functionalities. Thus, the hardware remains the same and the only cost increase is due to the increase in memory for these additional software modules.

The test setup, referring to Figure 1, is as follows. The Gatekeeper used is Cisco 2600 series, SIP UA1 and UA2 are Pingtel xpressa software evaluation versions, H.323 EP1 and EP2 used are NetMeeting and the phone used is a normal four –wire analog phone. The SIP Server and MGC are implemented on Linux for this experiment. The H.323 stack registers to the fixed or discovered gatekeeper and listens for the incoming call (SETUP) on port number 1720. SIP Stack can use TCP as well as UDP. In our experiment UDP is used. SIP stack registers to the SIP Server at port 5060 and listens for the incoming SIP call (INVITE) at port 5060. Also, Media Gateway (MG) registers to Media Gateway Controller (MGC) at port number 2427 and listens for the signals from MGC at port number 2427. The IP address is same for the registrations to the H.323 Gatekeeper, SIP server and MGC. The H.323 alias or SIP phone name or MG Endpoint name is set to aaln/1.

The E164 alias can be set to any number and possibly the same number can be used to register to the GK, SIP Registrar and MGC. If the same number is needed, we have to make sure that while the multi-protocol endpoint registers to the H.323 GK and the SIP server, it uses the same number assigned to the MG part by the MGC. Of course, this number has to be accepted by the GK and the SIP server. Otherwise the endpoint has to use three different numbers for the three protocols. In the implementation it is also made sure that there is no clash for the port numbers between the VoIP stacks.

The call control modules of the respective VoIP stacks call the stack APIs and start the instances for the respective stacks with appropriate initializations. The VoIP stacks are implemented in such a way that when the call control module starts the respective instances of the stack a pointer to the Call back function is passed to the stack. These respective call back functions receive all the messages and the call back data from the respective stacks. The call back messages for the three stacks are kept as similar as possible so that the implementation of the call back function is easy for the user.

Though we have separately explained the state machines for the different Call Control (CC) modules and the UI Control modules, it is essential to explain the message passing and interaction between these modules and with other modules during a call. This can be explained using the test cases described below. Consider an instance wherein there is an outgoing call from Endpoint A to MG B. To initiate the call at Endpoint A, the user presses the MGCP button and then goes off-hook. When the user goes off-hook, Line Interface detects the off-hook event and then informs it to the Codec Manager. The Codec Manager sends the off-hook message to the UI Control module. The UI module on receiving the off-hook event checks the call availability status. If there is a free call available, M_OFF_HOOK message is send to the MGCP CC module and the number of free calls available is decremented to zero. Also, since it is a MG call the dial tone is not played immediately to the user and there is no state change. The MGCP CC module pending for a message in the message queue receives the M_OFF_HOOK message and calls the MGCP stack API for sending the off-hook event (EVENT_HD) to the MGC. MGC on receiving the EVENT_HD message should respond with playing of dial tone signal (SIGNAL_DL) and enabling of DTMF detection. M_CB_DIALTONE message is send to the UI control module so that dial tone can be played to the user. When the UI Control receives the M_CB_DIALTONE, it changes state to the “Dialing” state and waits for the collection of the calling party phone number from the user. The user hears the dial tone and starts keying in the number. The DTMF events are detected by the DSP and sent via the message decoder to the UI Control module. When the user finishes keying in the number, which is indicated by the “#” key, PLACE_CALL message is sent to the MGCP Call control module and the state is changed to “Call” state. On receiving the PLACE_CALL message, the state of the CC module is changed to the “Call Establishment”. If the far end starts ringing, the CC module receives the SIGNAL_RT signal to play ring back tone and the state is changed to “Ring-back state”. On receiving SIGNAL_RT, CallEvent(rem_ringing) is sent to the UI Control. UI Control on receiving this message plays the ring-back tone. When the media negotiation is over using the Session Description Protocol, Media Start call back message for TX and RX is sent along with the remote IP address, port number for the RTP/RTCP session, the encoder/decoder vocoders type and the packetization size to open the DSP channels. On receiving call back messages for the media, Callevent(rem_connect) is sent to the UI Control module so that the ring back tone can be stopped. The Voice Manager starts RTP instances and a message is sent to open the DSP channels for particular encoder/decoder type. On receiving the acknowledgement from the DSP, ACK message is sent to the CC module. On receiving the ACK, CC module changes state, in sequence, to “Communication” state and the voice call is established. During the call, if there is a second incoming call from other H.323/SIP endpoint, Incoming Call message is sent to the UI Control and the state is changed to the “Wait Call Accept” state. When the UI Control receives the Incoming Call message, it checks the availability of a free call. As there is no free call available, TC_EVENT (Cancelled) is sent to the H.323/SIP CC module. On receiving this message, the state machine of the H.323/SIP CC module is reset to “Standby” state and the <Protocol>CallReject API is invoked. This API will send Release Complete in case of H.323 and Cancel message in case of SIP with reason as “Busy” in both the cases. If the second incoming call were for the MG endpoint, then anyway MGC would respond with a busy signal. After the voice conversation is over and the remote party disconnects, the CC module receives SIGNAL_DL and the RTP instances are stopped. The DSP channels are also closed and the state is changed to “Standby” state. This remote disconnect message is forwarded to the UI control module, where it changes state to the “Busy tone” and plays the busy tone to the user. After a time out, the state is changed to the “WaitOnHook” and when the user goes on-hook state is changed to “Standby” state.

In another test scenario, when there is an incoming call to H.323 or SIP terminal, and if the free call is available, M_OFF_HOOK_OTHER_CALL is sent to the MG CC module. On receiving this message, MG CC module sends EVENT_HD and changes state to “Busy State”. Now if there is a second incoming call for H.323/SIP, the call available is zero and thus the call is rejected. If the incoming call is for MG then since the off-hook event is already send to MGC, MGC responds with a busy signal. Our system makes sure that when one call is going on, then another call cannot be initiated by the user unless the on going call is disconnected. Hence we handle all the cases for the incoming/outgoing calls, simultaneously, for the three VoIP signaling protocols.

5 CONCLUSIONS

In this paper, a novel multi-protocol signaling endpoint architecture has been described. A detail of the

design of such an endpoint has also been given. This has been implemented and tested in a practical, real world environment. Further work is in progress to formulate, in such an endpoint, the state machines for the supplementary services like Call Hold, Call Waiting etc. This would give the endpoint more degrees of freedom for example, putting an ongoing call on hold and attending a call from a different signaling endpoint. It is also planned to test this solution at the interoperability events organized by different consortiums.

REFERENCES

[1] Jonathan Davidson and James Peters, "Voice over IP Fundamentals", *Cisco Press* 2000.

[2] Hong Liu et al, "Voice over IP signaling: H.323 and Beyond", *IEEE Communication Magazine*, October 2000, pg. 142-148.

[3] "H.323 – Packet based Multimedia Communication Systems", *ITU-T standard 09/99.*

[4] The SIP protocol stack *RFC 2543*, SIP: Session Initiation Protocol, March 1999.

[5] MGCP NCS 1.0 The PacketCable profile, "Network -Based Call Signaling (NCS) Protocol", *PKT-SP-EC-MGCP-I02-99120.*

[6] "Understanding Voice Packet Protocols", *White Paper*, Cisco Systems Inc.

[7] Daniele Rizzetto et al, "A voice over IP service Architecture for integrated Communication", *IEEE Networks*, May/June, 1999, pg. 34-40.

[8] Bill Douskalis, " IP Telephony – The Integration of Robust VoIP Services", *Prentice Hall PTR* 2000.

[9] Bur Goode, "Voice over Internet Protocol (VoIP)", *Proceedings of the IEEE*, Vol.90, No.9, September 2002, pg. 1495-1517

[10] Oliver Hersent et al, "IP Telephony – Packet based Multimedia Communication System", *Addison Wesley* 2000.

[11] Real-Time Protocol (RTP) *RFC 1889*, Audio-Video Transport Working Group, January 1996.

[12] SDP: The Session Description Protocol *RFC 2327*, April 1998.

[13] "Telephony Control for IP Phone and Little Gateway 2" Technical document by Cap Gemini Ernst & Young, in an Internal Communication to STMicroelectronics, 2001.