Design of a Call Connection Interface for VOIP Signaling between Two H.323 Terminals

T. Vrind, B. Siddharth and K. Ramathreya
Advisor Dr. P. K. Mahanti
Department of Computer Science and Engineering
Birla Institute of Technology, Mesra, Ranchi, India

INTRODUCTION

This project specifies the procedures for dynamically establishing, maintaining, and terminating call/connection between two H.323 terminals for real time transfer of data in the form of chat/mail/talk between the two communicating terminals by the help of a Gatekeeper. The main function of the Gatekeeper is to register the terminals with an alias name which is generally an E-mail id or an E-164 id (similar to a 10 digit phone number). The Gatekeeper is then involved in establishing the connection between the terminals if the called terminal and the calling terminal are both registered with the Gatekeeper. During the connected state, the two terminals can exchange voice packets, but since audio codecs are not supported on a UNIX/LINUX terminal, we have implemented a real time transfer of data in the form of messages entered by the users through the keyboard.
The project involved an extensive study of the H.323 standard, the various Registration Admission Status (RAS) messages, H.225 (Call Signaling) messages and H.245 (Control Signaling) messages and their implementation in the LINUX environment for establishment, maintenance and termination of connection between end points.
Since the development of the software was to be done in a single semester, the project was divided into four modules:

- Design of RAS, H.225 and H.245 message packets: This involved the identification of the various fields of each message and creating structures for each message. The fields of the messages are filled by the terminals and sent to the Gatekeeper to register, un-register, admission, locate-query & disengage.
- Design of the Gatekeeper: The Gatekeeper acts as a server and is implemented using a finite state machine which changes its state upon receiving/sending messages to the terminal which is communicating with it. In short it acts as a brain or a focal point of the H.323 network.
- Design of the terminals: Terminals form the backbone of the H.323 network. They act as clients to the Gatekeeper (server) and exchange messages in order to obtain a logical connection between themselves (the two terminals). A terminal is a finite state machine, which change state upon receiving and sending messages with the Gatekeeper and/or with another terminal.
- Design of the User Interface: The user interface has been developed for each terminal through which the user can access the network for communicating with

another terminal. This front end of the package makes it user friendly and abstracts the intricacies involved.

INTRODUCTION TO H.323 STANDARD

H.323 is a standard that specifies the components, protocols and procedures that provide multimedia communication services - real-time audio, video and data communications – over packet networks, including IP – based networks. H.323 is a part of a family of ITU – T recommendations called H.32x that provides multimedia communication services over a variety of networks. The H.323 standard is a cornerstone technology for the transmission of real – time audio,video and data communications over packet based networks. Packet based networks include IP - based networks or Internet packet exchange (IPX) – based Local Area Networks
(LANs), enterprise networks (Ens), metropolitan area networks (MANs) & wide area networks (WANs). H.323 can be applied in a variety of mechanisms - audio only (IP – telephony), audio and video (video telephony) : audio and data, and audio, video and data. H.323 can also be applied to multipoint - multimedia communications. It can be applied in a variety of areas – consumer, business, and entertainment applications.

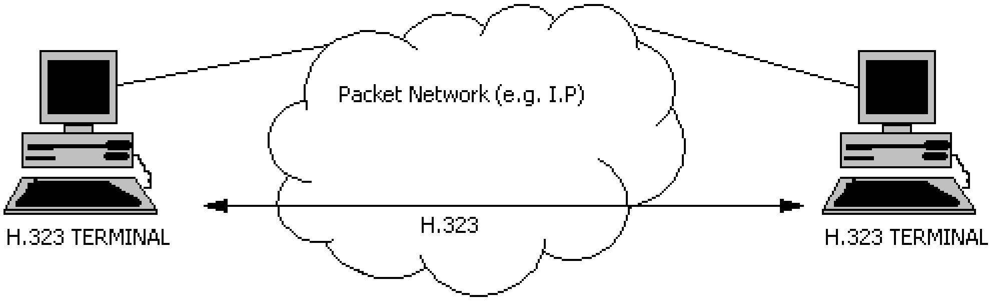

Figure 1: H.323 Terminals on a Packet Network

H.323 COMPONENTS

The H.323 standard specifies four kinds of components, which when networked together, provide the point to point and point to multimedia – communication services.

- Terminals
- Gateways
- Gatekeepers
- Multipoint Contol Units (MCUs)

Terminals

Used for real – time bi-directional multimedia communications, an H.323 terminal can either be a PC or a stand – alone device, running an H.323 and the multimedia applications. It supports audio communications and can optionally support video or data communications. Because the basic service provided by the H.323 terminal is audio communications, an H.323 terminal plays a key role in IP – telephony services. The primary goal of an H.323 is to interwork with other multimedia terminals. H.323 terminals are compatible with H.324 terminals on SCN and wireless netwoks, H.310 terminals on B-ISDN, H.320 terminals on ISDN ,H.321 terminals on B-ISDN , and H.322 terminals on guaranteed QOS LANs. H.323 terminals may be used in multipoints conferences.

Gateways

A Gateway connects two dissimilar networks. Gateway provides connectivity between an H.323 network and non – H.323 network. For example, a Gateway can provide and connect communication between an H.323 terminal and SCN network (SCN networks include all switched telephony networks, e.g. PSTN). The connectivity of dissimilar networks is achieved by translating protocols for call setup and release, converting media formats between different networks, transferring information between the networks connected by the gateway. A gateway is not required, however, for communication between two terminals on an H.323 network.

Gatekeepers

Gatekeeper can be considered the “brain” of an H.323 network. It is the focal point for all calls within the H.323 network. Although they are not required, they provide important services such as addressing, authorization, and authentication of terminals and gateways, bandwidth management, accounting, charging. Gatekeepers may also provide call – routing services.

Multipoint Control Units

MCUs provide support for conferences of three or more H.323 terminals. All terminals participating in the conference establish a connection with the MCU. The MCU manages conference resources, negotiates between terminals for the purpose of determining the audio or video codec to use, and may handle the media stream. The gatekeepers, gateways and MCUs though logically separate but are implemented as single physical device.

PROTOCOLS SPECIFIED BY H.323

The protocols are independent of the packet networks and the transport protocols over which it runs and does not specify them. The protocols specified by H.323 are listed below:

- Audio codecs
- Video codecs
- H.225 registration, admission and status (RAS)
- H.225 call signaling
- H.245 control signaling
- Real time transfer protocol (RTP)
- Real time control protocol (RTCP)

Audio Codec

An audio codec encodes the audio signal from the microphone for transmission on the transmitting H.323 terminal and decodes the received audio code that is send to the speaker on the receiving H.323 terminal. Because audio is the minimum service provided by the H.323 standard, all H.323 terminal must have atleast one audio codec support as specified by ITU – T G.711 recommendation (audio coding at 64 Kbps).

Video Codec

A video codec encodes video from the camera for transmission and decodes received video code that is sent to the video display on the receiving H.323 terminal. Support of video codecs is optional. Specifications are as per ITU – T H.261 recommendation.

H.225 Registration, Admission, Status

Registration, Admission and Status (RAS) is the protocol between endpoints (terminals and gateways) and gatekeepers. The RAS is used to perform registration, admission, bandwidth changes, status and disengage procedures between endpoints and gatekeepers. A RAS channel is used to exchange RAS messages. The signalling channel is opened between an endpoint and a gatekeeper prior to the establishment of any other channels.

H.225 Call Signaling

The H.225 call signaling is used to establish connection between two H.323 end points.

This is achieved by exchanging H.225 protocol messages on the call – signaling channel. The call signaling channel is opened between two H.323 end points or between an end point and the gatekeeper.

H.245 Control Signaling

H.245 control signaling is used to exchange end – to – end control messages governing the operation of the H.323 end point. These control messages carry information related to the following:

- Capabilities exchange
- Opening and closing of logical channels used to carry media streams
- Flow – control messages
- General command and indications

Real – Time Transport Protocol

Real Time Transport Protocol (RTP) provides end to end delivery services of real time audio and video. Whereas H.323 is used to transport data over IP – based networks, RTP is typically used to transport data via the User Datagram Protocol (UDP). RTP together with UDP, provides transport protocol functionality. RTP provides payload type identification, sequence numbering, time stamping, and delivery monitoring. UDP provides multiplexing and check sum services. RTP can also be used with other transport protocols.

Real – Time Transport Control Protocol

Real Time Transport Control Protocol (RTCP) is the counter part of RTP that provides control services. The primary function of RTCP is to provide feedback on the quality of the data distribution. Other RTCP functions include carrying a transport - level identifier for an RTP source called a canonical name, which is used by receivers to synchronize audio and video.

SYSTEM ARCHITECTURE

The proposed system consists of two terminals and one gatekeeper, which has to establish a point – to – point network between the two.

TERMINAL

The terminal is a finite state machine and has the following characteristics:

- H.245 for exchanging terminal capabilities and creation of media channels,
- H.225 for call signaling and call setup,
- RAS for registration and other admission control with a gatekeeper.

Although, an H.323 terminal must also support the G.711 audio codec and RTP/RTCP for sequencing audio and video packets, we could not support these two in our system because it was implemented in a LINUX environment.

GATEKEEPER

The gatekeeper designed for this system provide call control services for terminal end points, such as address translation, admission control and bandwidth control as defined within RAS a gatekeeper is important because it maintains a table for all the terminal registering with it with an alias name. The gatekeeper maintains a record of the transport address (IP address + Port No.), and the alias name for each new terminal rgistering with it. The gatekeeper translates the incoming E.164 telephone address or the Email Id (alias name) into transport addresses by looking into the record maintained by it. We have used the method of direct call signaling for establishing connection between the two terminals, in which the gatekeeper transfers the transport address of the called party to the calling party in order to establish a logical channel between the two terminals for direct communication. The various functions of the gatekeeper are:

- Address Translation.
- Admission Control.
- Bandwidth Control.
- Locate Queries.

OVERVIEW

The list of messages defined in the H.323 protocols and used in the system are listed below:

RAS MESSAGES:

Terminal and Gateway Discovery messages

The GRQ message requests that any gatekeeper receiving it respond with a GCF granting it permission to register. The GRJ is a rejection of this request indicating that the requesting endpoint should seek another gatekeeper.

GatekeeperRequest (GRQ)

Note that one GRQ is sent per logical endpoint; thus an MCU or a Gateway might send many.

The GRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

protocolIdentifier – Identifies the H.225.0 vintage of the sending endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

rasAddress – This is the transport address that this endpoint uses for registration and status messages.

endpointType – This specifies the type(s) of the endpoint that is registering (the MC bit shall not be set by itself).

gatekeeperIdentifier – String to identify the gatekeeper from which the terminal would like to receive permission to register. A missing or null string **gatekeeperIdentifier** indicates that the terminal is interested in any available gatekeeper.

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

endpointAlias – A list of alias addresses, by which other terminals may identify this terminal.

alternateEndpoints – A sequence of prioritized endpoint alternatives for rasAddress, endpointType, or endpointAlias.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

authenticationCapability – This indicates the authentication mechanisms supported by the endpoint.

algorithmOIDs –

integrity – Indicates to the recipient which integrity mechanism is to be applied on the RAS messages.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

GatekeeperConfirm (GCF)

The GCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the GRQ.

protocolIdentifier – Identifies the vintage of the accepting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

gatekeeperIdentifier – String to identify gatekeeper that is sending the GCF.

rasAddress – This is the transport address that the gatekeeper uses for registration and status messages.

alternateGatekeeper – Sequence of prioritized alternatives for gatekeeperIdentifier and rasAddress. The client should use these alternatives in the future, should a request to the gatekeeper not respond or return a reject without redirect.

authenticationMode – This indicates the authentication mechanism to be used. The gatekeeper must choose **authenticationMode** from **authenticationCapability** provided by the endpoint in GRQ.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

algorithmOID –

integrity – Indicates to the recipient which integrity mechanism is to be applied on the RAS messages.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

GatekeeperReject (GRJ)

The GRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the GRQ.

protocolIdentifier – Identifies the vintage of the rejecting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

gatekeeperIdentifier – String to identify gatekeeper that is sending the GRJ.

rejectReason – Codes for why the GRQ was rejected by this gatekeeper.

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Terminal and Gateway Registration messages

The RRQ is a request from a terminal to a gatekeeper to register. If the gatekeeper responds with a RCF, the terminal shall use the responding gatekeeper for future calls. If the gatekeeper responds with a RRJ, the terminal must seek another gatekeeper to register with.

RegistrationRequest (RRQ)

The RRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any response associated with this specific message.

protocolIdentifier – Identifies the H.225.0 vintage of the sending endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

discoveryComplete – Set to TRUE if the requesting endpoint has preceded this message with the gatekeeper discovery procedure; set to FALSE if registering only. Note that registration may age, and the endpoint will get a failure on an RRQ or ARQ with a reason code of **discoveryRequired** or **notRegistered** respectively. This indicates that the endpoint should perform the discovery procedure (either dynamic or static) before issuing the RRQ with **discoveryComplete** set to TRUE.

callSignalAddress – This is the call signalling transport address for this endpoint. If multiple transports are supported, they must be registered all at once.

rasAddress – This is the registration and status transport address for this endpoint.

terminalType – This specifies the type(s) of the endpoint that is(are) registering; note that the MC bit shall not be set by itself; either the terminal, MCU, gateway, or gatekeeper bit shall also be set. If vendor information is provided, this information shall be identical to that in **endpointVendor**.

terminalAlias -This optional value is a list of alias addresses, by which other terminals may identify this terminal. If the **terminalAlias** is null, or an E.164 address is not present, an E.164 address may be assigned by the gatekeeper, and included in the RCF. If an email-ID is available for the endpoint, it should be registered. Note that multiple alias addresses may refer to the same transport addresses. All of the endpoint's aliases shall be included in each RRQ.

gatekeeperIdentifier – String to identify the gatekeeper that the terminal wishes to register with.

endpointVendor – Information about the endpoint vendor.

alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress, rasAddress, terminalType, or terminalAlias.

timeToLive – Duration of the validity of the registration, in seconds. After this time the gatekeeper may consider the registration stale.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

keepAlive – If set to TRUE indicates that the endpoint has sent this RRQ as a "keep alive". An endpoint can send a lightweight RRQ consisting of only keepAlive, endpointIdentifier, gatekeeperIdentifier, tokens, and timeToLive. A gatekeeper in receipt of RRQ with a keepAlive field set to TRUE should ignore fields other than endpointIdentifier, gatekeeperIdentifier, tokens, and timeToLive.

endpointIdentifier – The endpointIdentifier provided by the gatekeeper during the original RCF.

willSupplyUUIEs – If set to TRUE, this indicates that the endpoint will supply Q.931 message information in IRR messages if requested by the gatekeeper.

RegistrationConfirm (RCF)

The RCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the RRQ.

protocolIdentifier – Identifies the vintage of the accepting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callSignalAddress – This is an array of transport addresses for H.225.0 call signalling messages; one for each transport that the gatekeeper will respond to. This address includes the TSAP identifier.

terminalAlias – This optional value is a list of alias addresses, by which other terminals may identify this terminal.

gatekeeperIdentifier – String to identify the gatekeeper that has accepted the terminals registration.

endpointIdentifier – A gatekeeper assigned terminal identity string; shall be echoed in subsequent RAS messages.

alternateGatekeeper – Sequence of prioritized alternateGatekeeper for gatekeeperIdentifer and rasAddress. The client should use these alternateGatekeeper in the future, should a request to the gatekeeper not respond or return a reject without redirect.

timeToLive – Duration of the validity of the registration, in seconds. After this time the gatekeeper may consider the registration stale.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

willRespondToIRR – True if the Gatekeeper will send an IACK or INAK message in response to an unsolicited IRR message with its **needsResponse** field set to TRUE.

preGrantedARQ – Indicates events for which the gatekeeper has pre-granted admission. This allows for faster call setup times in environments where admission is guaranteed through means other than the ARQ/ACF exchange. Note that even if these fields are set to TRUE, an endpoint can still send an ARQ to the gatekeeper for reasons such as address translation, or the endpoint does not support this modified signalling mode. If the **preGrantedARQ** sequence is not present, then ARQ signaling shall be used in all cases. The flags are:

> **makeCall –** If the **makeCall** flag is TRUE then the gatekeeper has pre-granted permission to the endpoint to initiate calls without first sending an ARQ. If the **makeCall** flag is FALSE, the endpoint shall always send ARQ to get permission to make a call.
>
> **useGKCallSignalAddressToMakeCall –** If the **makeCall** and **useGKCallSignalAddressToMakeCall** flags are both set to TRUE, then if the endpoint does not send an ARQ to the gatekeeper to make a call, the endpoint shall send all H.225 call signalling to the gatekeeper call signalling channel.
>
> **answerCall** – If the **answerCall** flag is TRUE then the gatekeeper has pre-granted permission to the endpoint to answer calls without first sending an ARQ. If the **answerCall** flag is FALSE, the endpoint shall always send ARQ to get permission to answer a call.
>
> **useGKCallSignalAddressToAnswer –** If the **answerCall** and **useGKCallSignalAddressToAnswer** flags are both set to true, then when an endpoint does not send an ARQ to the gatekeeper to answer a call, the endpoint shall ensure that all H.225.0 call signalling comes from the gatekeeper. If an endpoint has been instructed to use the gatekeeper when answering, but it does not know whether an incoming call has come from the gatekeeper (which may involve looking at the transport address), the endpoint shall issue ARQ irrespective of the state of the **useGKCallSignalAddressToAnswer** flag.

RegistrationReject (RRJ)

The RRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the RRQ.

protocolIdentifier – Identifies the vintage of the rejecting gatekeeper.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

rejectReason – The reason for the rejection of the registration.

gatekeeperIdentifier – String to identify the gatekeeper that has rejected the terminal's registration.

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Terminal/Gatekeeper Unregistration messages

UnregistrationRequest (URQ)

The URQ requests that the association between a terminal and a gatekeeper be broken. Note that unregister is bidirectional, i.e. a Gatekeeper can request a terminal to consider itself unregistered, and a terminal can inform a Gatekeeper that it is revoking a previous registration.
The URQ message includes the following:
requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any response associated with this specific message.
callSignalAddress – This is one or more of the transport call signalling addresses for this endpoint which are to be unregistered.
endpointAlias – This optional value is a list of alias addresses, by which other terminals may identify this terminal. If this optional field is not present, all aliases are unregistered in a single message. The E.164 address, if assigned, is required. Only values listed here are unregistered; this allows, for example, an H323_ID to be unregistered while leaving the E.164 address registered.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
endpointIdentifier – Confirmation of identity; not sent by the gatekeeper.
alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress or endpointAlias.
gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous URJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.
reason – Used when the gatekeeper sends the URQ to indicate why the gatekeeper considers the endpoint unregistered.

UnregistrationConfirm (UCF)

The UCF message includes the following:
requestSeqNum – This shall be the same value that was passed in the URQ.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

UnregistrationReject (URJ)

The URJ message includes the following:
requestSeqNum – This shall be the same value that was passed in the URQ.
rejectReason – The reason for the rejection of the unregistration.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
altGKInfo – Optional information about alternative gatekeepers.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Terminal to Gatekeeper Admission messages

The ARQ message requests that an endpoint be allowed access to the packet-based network by the gatekeeper, which either grants the request with an ACF or denies it with an ARJ.

AdmissionRequest (ARQ)

The ARQ message includes the following:
requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.
callType – Using this value, gatekeeper can attempt to determine "real" bandwidth usage. The default value is **pointToPoint** for all calls. It should be recognized that the call type

may change dynamically during the call and that the final call type may not be known when the ARQ is sent.

callModel – If **direct**, the endpoint is requesting the direct terminal to terminal call model. If **gatekeeperRouted**, the endpoint is requesting the gatekeeper mediated model. The gatekeeper is not required to comply with this request.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs. When sending the ARQ to answer a call, destinationInfo indicates the destination of the call (the answering endpoint).

destCallSignalAddress – Transport address used at the destination for call signalling.

destExtraCallInfo – Contains external addresses for multiple calls.

srcInfo – Sequence of alias addresses for the source endpoint, such as E.164 addresses or H323_IDs. When sending the ARQ to answer a call, srcInfo indicates the originator of the call.

srcCallSignalAddress – Transport address used at the source for call signalling.

bandWidth – The number of 100 bits requested for the bidirectional call. For example, a 128 kbit/s call would be signalled as a request for 256 kbit/s. The value refers only to the audio and video bit rate excluding headers and overhead.

callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the ARQ with a particular call.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

conferenceID – Unique conference identifier.

activeMC – If TRUE, the calling party has an active MC; otherwise FALSE.

answerCall – Used to indicate to a gatekeeper that a call is incoming.

canMapAlias – If set to TRUE indicates that if the resulting ACF contains **destinationInfo**, **destExtraCallInfo** and/or **remoteExtension** fields, the endpoint can copy this information to the **destinationAddress**, **destExtraCallInfo** and **remoteExtensionAddress** fields of the SETUP message respectively. If the GK would replace addressing information from the ARQ and **canMapAlias** is FALSE, then the gatekeeper should reject the ARQ.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

srcAlternatives – A sequence of prioritized source endpoint alternatives for srcInfo, srcCallSignalAddress, or rasAddress.

destAlternatives – A sequence of prioritized destination endpoint alternatives for destinationInfo or destCallSignalAddress.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous ARJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

transportQOS – An endpoint may use this to indicate its capability to reserve transport resources.

willSupplyUUIEs – If set to TRUE, this indicates that the endpoint will supply Q.931 message information in IRR messages if requested by the gatekeeper.

The TransportQOS structure includes the following:

endpointControlled – The endpoint will apply its own reservation mechanism.

gatekeeperControlled – The gatekeeper will perform resource reservation on behalf of the endpoint.

noControl – No resource reservation is needed.

NOTE – Both **destinationInfo** and **destCallSignalAddress** are not required, but at least one shall be present unless the endpoint is answering a call. There is no absolute rule over which is preferred as this may be site specific, but the E.164 address should be provided if available. It is cautioned that the best results will be obtained by considering the nature of the transport protocols in use.

AdmissionConfirm (ACF)

The ACF message includes the following:

requestSeqNum – This shall be the same value that was passed in the ARQ.

bandWidth – The allowed maximum bandwidth for the call; may be less than that requested.

callModel – Tells terminal whether call signalling sent on destCallSignalAddress goes to a gatekeeper or to a terminal. A value of **gatekeeperRouted** indicates that call signalling is being passed via the gatekeeper, while **direct** indicates that the endpoint-to-endpoint call mode is in use.

destCallSignalAddress – The transport address to which to send Q.931 call signalling, but may be an endpoint or gatekeeper address depending on the call model in use.

irrFrequency – The frequency, in seconds, that the endpoint shall send IRRs to the gatekeeper while on a call, including while on hold. If not present, the endpoint does not send IRRs while active on a call, and it is expected that the gatekeeper will poll the endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

destinationInfo – The address of the initial channel, used when calling through a gateway.

destExtraCallInfo – Needed to make possible additional channel calls, i.e. for a 2*64 kbit/s call on the WAN side. Shall only contain E.164 addresses and shall not contain the number of the initial channel.

destinationType – This specifies the type of the destination endpoint.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways.

alternateEndpoints – A sequence of prioritized endpoint alternatives for destCallSignalAddress or destinationInfo.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

TransportQOS – The gatekeeper may indicate to the endpoint where the responsibility lies for resource reservation. If the gatekeeper received a TransportQOS in ARQ, then it should include TransportQOS (possibly modified according to gatekeeper implementation) in ACF.

willRespondToIRR – TRUE if the Gatekeeper will send an IACK or INAK message in response to an unsolicited IRR message when the IRR's **needsResponse** field set to TRUE.

uuiesRequested – The gatekeeper may request the endpoint to notify the gatekeeper of H.225.0 call signalling messages that the endpoint sends or receives if the endpoint indicated this capability in the ARQ by setting **willSupplyUUIEs** to TRUE. **uuiesRequested** indicates the set of H.225.0 call signalling messages of which the endpoint shall notify the gatekeeper.

AdmissionReject (ARJ)

The ARJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the ARQ.

rejectReason – Reason the admission request was denied.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

callSignalAddress – This is the gatekeeper's call signalling address returned when the reject reason is routeCallToGatekeeper.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Terminal to gatekeeper requests for changes in bandwidth

The BRQ message requests that an endpoint be granted a changed packet-based network bandwidth allocation by the gatekeeper, which either grants the request with a BCF or denies it with a BRJ.

The gatekeeper may request that an endpoint raise or lower the bandwidth in use with a BRQ. If the request is to raise the rate, the endpoint may reply with either BRJ or BCF. If the request is for a lower rate, the endpoint shall reply with a BCF if the lower rate is supported, otherwise with BRJ.

BandwidthRequest (BRQ)

The BRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

conferenceID – ID of the call that is to have the bandwidth changed.

callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the BRQ with a particular call.

callType – Using this value, gatekeeper can attempt to determine "real" bandwidth usage.

bandWidth – The NEW number of 100 bits increments requested for the call. This is an absolute value that includes only audio and video bitstreams not counting headers and overhead.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous BRJ message. Used as a back-up if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

answeredCall – Set to TRUE to indicate that this party was the original destination (this party answered the call).

BandwidthConfirm (BCF)

The BCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the BRQ.

bandWidth – The maximum allowed at this time in increments of 100 bits.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

BandwidthReject (BRJ)

The BRJ message includes the following:

requestSeqNum – This shall be the same value that was passed in the BRQ.

rejectReason – The reason the change was rejected by the gatekeeper.

allowedBandWidth – The maximum allowed at this time in increments of 100 bits including the current allocation.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

altGKInfo – Optional information about alternative gatekeepers.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Location Request messages

The LRQ requests that a gatekeeper provide address translation. The gatekeeper responds with an LCF containing the transport address of the destination, or rejects the request with LRJ.

LocationRequest (LRQ)

The LRQ message includes the following:

requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

replyAddress – Transport address to which to send the LCF/LRQ.

sourceInfo – Indicates the sender of the LRQ. The gatekeeper can use this information to decide how to respond to the LRQ.

canMapAlias – If set to TRUE indicates that if the resulting LCF contains **destinationInfo**, **destExtraCallInfo** and/or **remoteExtension** fields, the endpoint can copy this information to the **destinationAddress**, **destExtraCallInfo** and **remoteExtensionAddress** fields of the SETUP message respectively. If the GK would replace addressing information from the LRQ and **canMapAlias** is FALSE, then the gatekeeper should reject the LRQ.

gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous LRJ message. Used as a backup if the original gatekeeper did not respond or rejected the request.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

LocationConfirm (LCF)

The LCF message includes the following:

requestSeqNum – This shall be the same value that was passed in the LRQ.

callSignalAddress – The transport address to which to send Q.931 call signalling; uses the reliable well known or dynamic port, but may be an endpoint or gatekeeper address depending on the call model in use.

rasAddress – Registration, admissions, and status address for the located endpoint.

nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).

destinationInfo – Sequence of alias addresses for the destination, such as E.164 addresses or H323_IDs.

destExtraCallInfo – Contains external addresses for multiple calls.

destinationType – This specifies the type of the destination endpoint.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways.
alternateEndpoints – A sequence of prioritized endpoint alternatives for callSignalAddress, rasAddress, or destinationInfo.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

LocationReject (LRJ)

The LRJ message includes the following:
requestSeqNum – This shall be the same value that was passed in the LRQ.
rejectReason – Reason the location request was denied.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
altGKInfo – Optional information about alternative gatekeepers.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

Disengage messages

DisengageRequest (DRQ)

If sent from an endpoint to a gatekeeper, the DRQ informs the gatekeeper that an endpoint is being dropped. If sent from a gatekeeper to an endpoint, the DRQ forces a call to be dropped; such a request shall not be refused. The DRQ is not sent between endpoints directly.
Note that DRQ is not the same as **ReleaseComplete** since its purpose is to inform the gatekeeper of the termination of a call; the gatekeeper may not receive the release complete if it is not terminating the call signalling channel.
The DRQ message includes the following:
requestSeqNum – This is a monotonically increasing number unique to the sender. It shall be returned by the receiver in any messages associated with this specific message.

endpointIdentifier – This is an endpoint identifier that was assigned to the terminal by RCF.
conference ID – ID of the call that is to have the bandwidth released.
callReferenceValue – The CRV from Q.931 for this call; only local validity. This is used by a gatekeeper to associate the message with a particular call.
disengageReason – The reason the change was requested by the gatekeeper or the terminal.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.
gatekeeperIdentifier – A gatekeeperIdentifier which the client received in the alternateGatekeeper list in RCF from the gatekeeper when it registered or in a previous DRJ message. Used as a back-up if the original gatekeeper did not respond or rejected the request.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.
answeredCall – Set to TRUE to indicate that this party was the original destination (this party answered the call).

DisengageConfirm (DCF)

The DCF message includes the following:
requestSeqNum – This shall be the same value that was passed in the DRQ.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

DisengageReject (DRJ)

DRJ is sent by the gatekeeper if the endpoint is unregistered.

The DRJ message includes the following:
requestSeqNum – This shall be the same value that was passed in the DRQ.
rejectReason – The reason the request was rejected.
nonStandardData – Carries information not defined in this Recommendation (for example, proprietary data).
altGKInfo – Optional information about alternative gatekeepers.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
integrityCheckValue – Provides improved message integrity/message authentication of the RAS messages. The cryptographically based integrity check value is computed by the sender applying a negotiated integrity algorithm and the secret key upon the entire message. Prior to integrityCheckValue computation, this field shall be ignored and shall be empty. After computation, the sender puts the computed integrity check value in the integrityCheckValue field and transmits the message.

H.225 CALL SIGNALING (Q.931 specifications):

Alerting

This message may be sent by the called user to indicate that called user alerting has been initiated. In everyday terms, the "phone is ringing."

H.225.0 – Alerting

Information element	**H.225.0 status (M/F/O)**	**Length in H.225.0**
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
Signal	O	2-3

High layer compatibility	FFS	NA
User-to-User	M	2-131

The user-to-user information element contains the Alerting-UUIE defined in the H.225.0 Message Syntax. The Alerting-UUIE includes the following:
protocolIdentifier – Set to the version of H.225.0 supported.
destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.
h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling. This address may also be sent in Call Proceeding, Progress, or Connect.
callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.
h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect**.**
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

Call Proceeding

This message may be sent by the called user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted.

H.225.0 – Call Proceeding

Information element	**H.225.0 status (M/F/O)**	**Length in H.225.0**
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*

Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
High layer compatibility	FFS	NA
User-to-User	M	2-131

The user-to-user information element contains the CallProceeding-UUIE defined in the H.225.0 Message Syntax. The CallProceeding-UUIE includes the following:
protocolIdentifier – Set to the version of H.225.0 supported.
destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.
h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling.
callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.
h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect**.**
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

Connect

This message shall be sent by the called entity to the calling entity (gatekeeper, gateway, or calling terminal) to indicate acceptance of the call by the called entity.

H.225.0 – Connect

Information element	**H.225.0 status (M/F/O)**	**Length in H.225.0**
Protocol discriminator	M	1
Call reference	M	3
Message type	M	1
Bearer capability	O (Note)	5-6

Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	O	2-4
Notification indicator	O	2-*
Display	O	2-82
Date/Time	O	8
High layer compatibility	FFS	NA
Low layer compatibility	FFS	NA
User-to-User	M	2-131
NOTE – Bearer capability is mandatory if the message is between a terminal and a gateway.		

The user-to-user information element contains the Connect-UUIE defined in the H.225.0 Message Syntax. The Connect-UUIE includes the following:

protocolIdentifier – Set by the called endpoint to the version of H.225 supported.

h245Address – This is a specific transport address on which the called endpoint or gatekeeper handling the call would like to establish H.245 signalling. This address shall be sent if sent earlier in Alerting, Progress, or Call Proceeding.

destinationInfo – Contains an **EndpointType** to allow the caller to determine whether the call involves a gateway or not.

conferenceID – Will contain a unique number to allow the conference to be uniquely identified from all others as received in the Setup.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityMode – An H.323 entity that receives a Setup message with the **h245SecurityCapability** set shall respond with the corresponding, acceptable **h245SecurityMode** in the CallProceeding, Alerting, Progress, or Connect.

tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.

cryptoTokens – Encrypted **tokens**.

fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.

Setup

This message shall be sent by a calling H.323 entity to indicate its desire to set up a connection to the called entity.

H.225.0 – Setup

Information element	**H.225.0 status(M/F/O/CM)**	**Length in H.225.0**
Protocol discriminator	M	1
Call reference	M (Note 2)	3
Message type	M	1
Sending complete	O	1
Repeat indicator	F	NA
Bearer capability	M	5-6
Extended facility	O	8-*
Channel identification	FFS	NA
Facility	O	8-*
Progress indicator	F	NA
Network specific facilities	F	NA
Notification indicator	O	2-*
Display	O	2-82
Keypad facility	O	2-34
Signal	O	2-3
Calling party number	O	2-131
Calling party subaddress	CM (Note 1)	NA
Called party number	O	2-131
Called party subaddress	CM (Note 1)	NA
Transit network selection	F	NA
Repeat indicator	F	NA
Low layer compatibility	FFS	NA
High layer compatibility	FFS	NA

Information element	**H.225.0 status(M/F/O/CM)**	**Length in H.225.0**
User-to-User	M	2-131
NOTE 1 – Subaddresses are needed for some SCN call scenarios; they should not be used for packet-based network side only calls. NOTE 2 – If an ARQ was previously sent, the CRV used here shall be the same.		

The user-to-user information element contains the Setup-UUIE defined in the H.225.0 Message Syntax. The Setup-UUIE includes the following:

protocolIdentifier – Set to the version of H.225.0 supported.

h245Address – This is a specific transport address on which the calling endpoint or gatekeeper handling the call would like to establish the H.245 signalling. This should only be provided by the sender if it is capable of handling H.245 procedures before receiving a CONNECT on the Call Signalling channel.

sourceAddress – Contains the alias addresses for the source; the E.164 number of the source is in the Q.931 Calling Party Number IE**.** The primary address shall be first.

sourceInfo – Contains an **EndpointType** to allow the called party to determine whether the call involves a gateway or not.

destinationAddress – This is the address to which the endpoint wishes to be connected. The primary address shall be first. When calling an endpoint using only an E.164 address, this address shall be placed in the Q.931 Called Party Number IE. The destinationAddress, if available, shall be included in the Setup message by version 2 terminals.

destCallSignalAddress – Needed to inform the gatekeeper of the destination terminal's call signalling transport address; redundant in the direct terminal-to-terminal case. In all cases where the information is available to the sender of the Setup message, this field shall be filled in.

destExtraCallInfo – Needed to make possible additional channel calls, i.e. for a 2*64 kbit/s call on the WAN side. Shall only contain E.164 addresses and shall not contain the number of the initial channel. (See Note.)

destExtraCRV – CRVs for the additional SCN calls specified by **destExtraCallInfo.** Their use is for further study. They can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

activeMC – Indicates that the calling endpoint is under the influence of an active MC.

conferenceID – Unique conference identifier.

conferenceGoal:

> **create** – Start a new conference.
>
> **invite** – Invite a party into an existing conference.
>
> **join** – Join an existing conference.
>
> **capability-negotiation** – Negotiate capabilities for a later loosely-coupled conference.

callIndependentSupplementaryService – Transport of supplementary services APDUs in a non-call related manner.

callServices – Provides information on support of optional Q-series protocols to gatekeeper and called terminal.

callType – Using this value, called party's gatekeeper can attempt to determine 'real' bandwidth usage. The default value is **pointToPoint** for all calls; it should be recognized that the call type may change dynamically during the call and that the final call type may not be known when the Setup is sent.

sourceCallSignalAddress – Contains the transport address for the source; this value shall be used in the ARQ message by the receiver of the Setup. In all cases where the information is available to the sender of the Setup message, this field shall be filled in. The value of sourceCallSignalAddress shall be equal to the value that was used in the ARQ by the sender of the Setup, and shall be echoed by the endpoint receiving the Setup in its ARQ.

remoteExtensionAddress – Contains the alias address of a called endpoint in cases where this information is needed to traverse multiple Gateways. In all cases where the information is available to the sender of the Setup message, this field shall be filled in.

callIdentifier – A globally unique call identifier set by the originating endpoint which can be used to associate RAS signalling with the modified Q.931 signalling used in this Recommendation.

h245SecurityCapability – A set of capabilities the sender can use to secure the H.245 channel.
tokens – This is some data which may be required to allow the operation. The data shall be inserted into the message if available.
cryptoTokens – Encrypted **tokens**.
fastStart – Used only in the fast connect procedure, **fastStart** supports the signalling needed to open a logical channel. This uses the OpenLogicalChannel structure defined in Recommendation H.245, but the sender of this indicates the modes it prefers to receive and transmit, and the transport addresses where it expects to receive media streams.
mediaWaitForConnect – If TRUE, indicates that the recipient of the Setup message shall not transmit media until sending the Connect message.
canOverlapSend – If TRUE, indicates that the sender of Setup shall support overlap sending.

H.245 CONTROL SIGNALING:

Terminal Capability Set

This message contains information about the terminal's capability to transmit and receive. It also indicates the version of this Recommendation that is in use. It is sent from an outgoing CESE to a peer incoming CESE.
sequenceNumber is used to label instances of TerminalCapabilitySet so that the corresponding response can be identified.
protocolIdentifier is used to indicate the version of this Recommendation that is in use. Annex A lists the object identifiers defined for use by this Recommendation.
multiplexCapability indicates capabilities relating to multiplexing and network adaptation. A terminal shall include multiplexCapability in the first TerminalCapabilitySet sent.
V75Capability indicates the capabilities of the V.75 control entity. The audioHeader indicates the capability of the V.75 audio header.

Terminal Capability Set Acknowledge

This is used to confirm receipt of a TerminalCapabilitySet from the peer CESE.
The sequenceNumber shall be the same as the sequenceNumber in the TerminalCapabilitySet for which this is the confirmation.

Open Logical Channel

This is used to attempt to open a uni-directional logical channel connection between an outgoing LCSE and a peer incoming LCSE and to open a bi-directional logical channel connection between an outgoing B-LCSE and a peer incoming B-LCSE.

forwardLogicalChannelNumber: indicates the logical channel number of the forward logical channel that is to be opened.

forwardLogicalChannelParameters: include parameters associated with the logical channel in the case of attempting to open a uni-directional channel and parameters associated with the forward logical channel in the case of attempting to open a bi-directional channel.

reverseLogicalChannelParameters: include parameters associated with the reverse logical channel in the case of attempting to open a bi-directional channel. Its presence indicates that the request is for a bi-directional logical channel with the stated parameters, and its absence indicates that the request is for a uni-directional logical channel.

portNumber is a user to user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the logical channel.

dataType indicates the data that is to be carried on the logical channel.

If it is nullData, the logical channel will not be used for the transport of elementary stream data, but only for adaptation layer information – if video is to be transmitted in one direction only, but a retransmission protocol is to be used, such as AL3 defined in Recommendation H.223, a return channel is needed to transport the retransmission requests – it may also be used to describe a logical channel that only contains PCR values in the case of H.222.1 Transport Streams [9].

Terminals capable only of uni-directional (transmit or receive) operation on media types which make use of bi-directional channels shall send capabilities only for the supported direction of operation. The reverse direction shall use the nullData type, for which no capability is necessary. Transmit-only terminals should send transmit capabilities, but terminals should not assume that the absence of transmit capabilities implies that transmit-only operation is not possible.

separateStack indicates that a separate transport stack will be used to transport the data and provides an address to use to establish the stack which is either a Q.2931, E.164, or local area network transport address.

networkAccessParameters define the distribution, network address, and creation and association information to be used for the separateStack.

distribution shall be present when networkAddress is set to localAreaNetwork and shall indicate whether the networkAddress is a uni or multicast transport address.

networkAddress indicates the address of the actual stack in use: Q.2931, E.164, or local area network transport address.

associateConference indicates whether or not the data conference is new (associateConference=FALSE) or is an existing data conference which should be associated with the audio/video call (associateConference=TRUE).

externalReference indicates information which may be used to further provide association or information concerning the separateStack.

If it is of type VideoCapability, AudioCapability, the logical channel may be used for any of the variations indicated by each individual capability; and it shall be possible to switch between these variations using only signalling that is in-band to the logical channel – for example, in the case of H.261 video, if both QCIF and CIF are indicated, it shall be possible to switch between these on a picture-by-picture basis. In the case of DataApplicationCapability, only one instance of a capability can be indicated since there is no in-band signalling allowing a switch between variations.

If it is encryptionData, the logical channel will be used for the transport of encryption information as specified.

forwardLogicalChannelDependency indicates which logical channel number the forward channel to be opened is dependent on.

reverseLogicalChannelDependency indicates which logical channel number the reverse channel to be opened is dependent on.

The replacementFor parameter indicates that the logical channel to be opened will be a *replacement for* the specified existing, already-open logical channel. This parameter shall be used only to refer to logical channels already in the ESTABLISHED state. Logical channels opened using this parameter shall not carry any data traffic until after all traffic on the referenced established logical channel ceases. Media decoders will in this case never be required to decode data traffic from both logical channels simultaneously. Once traffic on the newly established logical channel has begun, the old logical channel shall immediately be closed. Receivers may acknowledge logical channels opened using the replacementFor mechanism with the understanding that the old and new logical channels shall not be used simultaneously, and therefore will not exceed the receiver's capability to decode. The encryptionSync field shall be used by the master when acknowledging the opening of a channel by a slave. It is used to provide the encryption key value and the synchronization point at which the key should be used. For H.323, the syncFlag shall be set to the RTP dynamic payload number which matches the key.

H222LogicalChannelParameters: is used to indicate parameters specific to using H.222.1 [9]. It shall be present in forwardLogicalChannelParameters and shall not be present in reverseLogicalChannelParameters.

resourceID indicates in which ATM Virtual Channel the logical channel is to be transported. The means by which this parameter is associated with an ATM Virtual Channel is not specified in this Recommendation.

subChannelID indicates which H.222.1 subchannel is used for the logical channel. It shall be equal to the PID in a Transport Stream and the stream_id in a Program Stream.

pcr-pid indicates the PID used for the transport of Program Clock References when the Transport Stream is used. It shall be present when the ATM virtual channel carries a Transport Stream and shall not be present when the ATM virtual channel carries a Program Stream.

programDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the program that the information to be carried in the logical channel is a part of.

streamDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the information that is to be carried in the logical channel.

H223LogicalChannelParameters: is used to indicate parameters specific to using H.223 [10]. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

adaptationLayerType indicates which adaptation layer and options will be used on the logical channel. The codepoints are as follows: nonStandard, al1Framed (AL1 framed mode), al1NotFramed (AL1 unframed mode), al2WithoutSequenceNumbers (AL2 with no sequence numbers present), al2WithSequenceNumbers (AL2 with sequence numbers present), and al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_S, that will be used, the size being measured in octets), al1M (AL1M defined in Annex C with the specified parameters), al2M (AL2M defined in Annex C with the specified parameters) or al3M (AL3M defined in Annex C with the specified parameters).

segmentableFlag, when equal to true indicates that the channel is designated to be segmentable, and when equal to false indicates that the channel is designated to be non-segmentable.

H223AL1MParameters: is used to indicate parameters specific to using adaptation Layer AL1M.

transferMode indicates whether framed mode or unframed mode is used.

headerFEC indicates whether FEC is SEBCH(16,7) or Golay(24,12).

The length of CRC bits for the payload is indicted by crcLength as 4, 12, 20 or 28 bits.

rcpcCodeRate indicates the RCPC code rate as 8/8, 8/9, ..., 8/32.

ArqType indicates the ARQ mode of operation: noARQ indicates no retransmission, typeIArq indicates ARQ type I, and typeIIArq indicates ARQ type II.

alpduInterleaving, if true, indicates the use of AL-PDU interleaving.

alsduSplitting, if true, indicates the use of AL-SDU splitting mode.

H223AL2MParameters: is used to indicate parameters specific to using adaptation Layer AL2M.

headerFEC indicates whether FEC is SEBCH(16,5) or Golay(24,12).

alpduInterleaving, if true, indicates the use of AL-PDU interleaving.

H223AL3MParameters: is used to indicate parameters specific to using adaptation Layer AL3M.

This has the same parameters as AL1MParameters, except transferMode and alsduSplitting are not present.

H2250LogicalChannelParameters: is used to indicate parameters specific to using H.225.0. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

The sessionID is a unique RTP or T.120 Session Identifier in the conference. It is used by the transmitter to refer to the session to which the logical channel applies. Only the master can create the session identification. By convention, there are three primary sessions. The first primary session with a session identification of 1 is the audio session, the second primary session with a session identification of 2 is the video session, and the third primary session with a session identification of 3 is the data session. A slave entity can open an additional session by providing a session identification of 0 in the openLogicalChannel message. The master will create a unique session identification and provide it in the openLogicalChannelAck message.

The associatedSessionID is used to associate one session with another. Typical use will be to associate an audio session with a video session to indicate which sessions to process for lip synchronization.

The mediaChannel indicates a transportAddress to be used for the logical channel. It is not present in the OpenLogicalChannel message when the transport is unicast. If the transportAddress is multicast, the master is responsible for creating the multicast transport address and shall include the address in the OpenLogicalChannel message. A slave entity that wishes to open a new multicast channel will provide zeroes in the multicast transportAddress field. The master will create and provide the multicast transportAddress in the OpenLogicalChannelAck message for the slave entity. Note that the MC will use the communicationModeCommand to specify the details about all the RTP Sessions in the conference.

The mediaChannel is used to describe the transport address for the logical channel. IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the "130" being encoded in the first octet of the OCTET STRING, followed by the "1" and so forth. The IPv6 address a148:2:3:4:a:b:c:d shall have the "a1" encoded in the first octet, "48" in the second, "00" in the third, "02" in the fourth and so forth. IPX addresses, node, netnum, and port shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

mediaGuaranteedDelivery indicates whether or not the underlying media transport should be selected to provide or not provide guaranteed delivery of data.

mediaControlChannel indicates the media control channel in which the sender of the open logical channel will be listening for media control messages for this session. This field is present only when a media control channel is required.

mediaControlGuaranteedDelivery indicates whether or not the underlying media control transport should be selected to provide or not provide guaranteed delivery of data. This field is present only when a media control channel is required.

The silenceSuppression is used to indicate whether the transmitter stops sending packets during times of silence. It shall be included in the openLogicalChannel message for an audio channel and omitted for any other type of channel.

destination indicates the terminalLabel of the destination if one has been assigned.

dynamicRTPPayloadType indicates a dynamic payload value which is used in H.323 for the H.225.0 alternative H.261 video packetization scheme. This field is present only when a dynamic RTP payload is in use.

mediaPacketization indicates which optional media packetization scheme is in use.

redundancyEncoding indicates that the redundant encoding method indicated in this parameter is to be used for the logical channel to be opened. The primary encoding is defined by the *dataType* of the *forwardLogicalChannelParameters* or the *reverseLogicalChannelParameters*, respectively. The type of redundancy encoding to be applied for this logical channel is identified by the *redundancyEncodingMethod* parameter, the secondary encoding is specified in the *secondaryEncoding* parameter. The *DataType* (audio, video, etc.) selected for both primary and secondary encoding shall match and shall be in accordance with the *redundancyEncodingMethod* selected.

The source parameter is used to identify the terminal number of the sender of the OpenLogicalChannel message.

h235 Key: is used to include, and specify the method by which media specific session keys are protected as they are passed between two endpoints. The encoding of this field is a nested ASN.1 value as described in Recommendation H.235.

EscrowData is used to specify the type and contents of any key escrow mechanism in use. Specific types and contents may be required by implementations when media encryption is enabled.

T120SetupProcedure indicates how the T.120 conference is to be set up. For originateCall and waitForCall, the caller should derive the T.120 numeric conference name from the H.323 CID (as described in Recommendation H.323), and issue the appropriate PDU (if the endpoint is master, it should issue an invite request, while a slave should issue a join request). For issueQuery, the caller should first issue a query request, and then set up the T.120 conference in accordance with the contents of the query response (as described in Recommendation T.124).

Open Logical Channel Acknowledge

This is used to confirm acceptance of the logical channel connection request from the peer LCSE or B-LCSE. In the case of a request for a uni-directional logical channel, it indicates acceptance of that uni-directional logical channel. In the case of a request for a bi-directional logical channel, it indicates acceptance of that bi-directional logical channel, and indicates the appropriate parameters of the reverse channel.

forwardLogicalChannelNumber indicates the logical channel number of the forward channel that is being opened.

reverseLogicalChannelParameters is present if and only if responding to a bi-directional channel request.

reverseLogicalChannelNumber indicates the logical channel number of the reverse channel.

portNumber is a user to user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the reverse logical channel.

multiplexParameters indicate parameters specific to the multiplex, H.222, H.223, or H.225.0, that is used to transport the reverse logical channel.

FlowControlToZero indicates whether the transmitter is allowed to start transmitting on the logical channel. If set to true, it indicates that the transmitter should not transmit on the logical channel until receiving a subsequent FlowControl message, applying to the logical channel, allowing it to do so. If set to false, or absent, the transmitter is allowed to begin transmitting immediately the channel is established.

The replacementFor parameter indicates that the logical channel to be opened will be a *replacement for* the specified existing, already-open logical channel. This parameter shall be used only to refer to logical channels already in the ESTABLISHED state. Logical channels opened using this parameter shall not carry any data traffic until after all traffic on the referenced established logical channel ceases. Media decoders will in this case never be required to decode data traffic from both logical channels simultaneously. Once traffic on the newly established logical channel has begun, the old logical channel shall immediately be closed. Receivers may acknowledge logical channels opened using the replacementFor mechanism with the understanding that the old and new logical channels shall not be used simultaneously, and therefore will not exceed the receiver's capability to decode.

separateStack indicates that a separate transport stack will be used to transport the data and provides an address, to use to establish the stack, which is either a Q.2931, E.164, or local area network transport address.

forwardMultiplexAckParameters indicate parameters specific to the multiplex, H.222, H.223, or H.225.0 that is used to transport the forward logical channel.

The encryptionSync field shall be used by the master when acknowledging the opening of a channel by a slave. It is used to provide the encryption key value and the synchronization point at which the key should be used. For H.323, the syncFlag shall be set to the RTP dynamic payload number which matches the key.

H2250LogicalChannelAckParameters are used to indicate parameters specific to using H.225.0.

sessionID is a unique RTP Session Identifier in the conference that can only be created by the master. It is created and provided by the master if the slave wishes to create a new session by specifying an invalid session identification of 0 in the openLogicalChannelAck message.

The mediaChannel indicates a transportAddress to be used for the logical channel. It shall be present in the OpenLogicalChannelAck message when the transport is unicast. If the transportAddress is multicast, the master is responsible for creating the multicast transport address and shall include the address in the OpenLogicalChannel message. A slave entity that wishes to open a new multicast channel will provide zeroes in the multicast transportAddress field. The master will create and provide the multicast transportAddress in the OpenLogicalChannelAck message for the slave entity. Note that the MC will use the communicationModeCommand to specify the details about all the RTP Sessions in the conference.

The mediaChannel is used to describe the transport address for the logical channel. IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the "130" being encoded in the first octet of the OCTET STRING, followed by the "1" and so forth. The IPv6 address a148:2:3:4:a:b:c:d shall have the "a1" encoded in the first octet, "48" in the second, "00" in the third, "02" in the fourth and so forth. IPX addresses, node, netnum, and port shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

mediaControlChannel indicates the media control channel in which the sender of the openLogicalChannelAck will be listening for media control messages for this session. This field is present only when a media control channel is required.

dynamicRTPPayloadType indicates a dynamic payload value which is used in H.323 for the H.225.0 alternative H.261 video packetization scheme. This field is present only when a dynamic RTP payload is in use.

portNumber is a user to user parameter that may be used by a user for such purposes as associating an input or output port, or higher layer channel number, with the logical channel.

dataType indicates the data that is to be carried on the logical channel.

If it is nullData, the logical channel will not be used for the transport of elementary stream data, but only for adaptation layer information – if video is to be transmitted in one direction only, but a retransmission protocol is to be used, such as AL3 defined in Recommendation H.223, a return channel is needed to transport the retransmission requests – it may also be used to describe a logical channel that only contains PCR values in the case of H.222.1 Transport Streams [9].

Terminals capable only of uni-directional (transmit or receive) operation on media types which make use of bi-directional channels shall send capabilities only for the supported direction of operation. The reverse direction shall use the nullData type, for which no capability is necessary. Transmit-only terminals should send transmit capabilities, but terminals should not assume that the absence of transmit capabilities implies that transmit-only operation is not possible.

separateStack indicates that a separate transport stack will be used to transport the data and provides an address to use to establish the stack which is either a Q.2931, E.164, or local area network transport address.

networkAccessParameters define the distribution, network address, and creation and association information to be used for the separateStack.

distribution shall be present when networkAddress is set to localAreaNetwork and shall indicate whether the networkAddress is a uni or multicast transport address.

networkAddress indicates the address of the actual stack in use: Q.2931, E.164, or local area network transport address.

associateConference indicates whether or not the data conference is new (associateConference=FALSE) or is an existing data conference which should be associated with the audio/video call (associateConference=TRUE).

externalReference indicates information which may be used to further provide association or information concerning the separateStack.

If it is of type VideoCapability, AudioCapability, the logical channel may be used for any of the variations indicated by each individual capability; and it shall be possible to switch between these variations using only signalling that is in-band to the logical channel – for example, in the case of H.261 video, if both QCIF and CIF are indicated, it shall be possible to switch between these on a picture-by-picture basis. In the case of DataApplicationCapability, only one instance of a capability can be indicated since there is no in-band signalling allowing a switch between variations.

If it is encryptionData, the logical channel will be used for the transport of encryption information as specified.

forwardLogicalChannelDependency indicates which logical channel number the forward channel to be opened is dependent on.

reverseLogicalChannelDependency indicates which logical channel number the reverse channel to be opened is dependent on.

The replacementFor parameter indicates that the logical channel to be opened will be a *replacement for* the specified existing, already-open logical channel. This parameter shall be used only to refer to logical channels already in the ESTABLISHED state. Logical channels opened using this parameter shall not carry any data traffic until after all traffic on the referenced established logical channel ceases. Media decoders will in this case never be required to decode data traffic from both logical channels simultaneously. Once traffic on the newly established logical channel has begun, the old logical channel shall immediately be closed. Receivers may acknowledge logical channels opened using the replacementFor mechanism with the understanding that the old and new logical channels shall not be used simultaneously, and therefore will not exceed the receiver's capability to decode.

The encryptionSync field shall be used by the master when acknowledging the opening of a channel by a slave. It is used to provide the encryption key value and the synchronization point at which the key should be used. For H.323, the syncFlag shall be set to the RTP dynamic payload number which matches the key.

H222LogicalChannelParameters: is used to indicate parameters specific to using H.222.1 [9]. It shall be present in forwardLogicalChannelParameters and shall not be present in reverseLogicalChannelParameters.

resourceID indicates in which ATM Virtual Channel the logical channel is to be transported. The means by which this parameter is associated with an ATM Virtual Channel is not specified in this Recommendation.

subChannelID indicates which H.222.1 subchannel is used for the logical channel. It shall be equal to the PID in a Transport Stream and the stream_id in a Program Stream.

pcr-pid indicates the PID used for the transport of Program Clock References when the Transport Stream is used. It shall be present when the ATM virtual channel carries a Transport Stream and shall not be present when the ATM virtual channel carries a Program Stream.

programDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the program that the information to be carried in the logical channel is a part of.

streamDescriptors is an optional octet string, which, if present, contains one or more descriptors, as specified in Recommendations H.222.0 and H.222.1, that describe the information that is to be carried in the logical channel.

H223LogicalChannelParameters: is used to indicate parameters specific to using H.223 [10]. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

adaptationLayerType indicates which adaptation layer and options will be used on the logical channel. The codepoints are as follows: nonStandard, al1Framed (AL1 framed mode), al1NotFramed (AL1 unframed mode), al2WithoutSequenceNumbers (AL2 with no sequence numbers present), al2WithSequenceNumbers (AL2 with sequence numbers present), and al3 (AL3, indicating the number of control field octets that will be present and the size of the send buffer, B_S, that will be used, the size being measured in octets), al1M (AL1M defined in Annex C with the specified parameters), al2M (AL2M defined in Annex C with the specified parameters) or al3M (AL3M defined in Annex C with the specified parameters).

segmentableFlag, when equal to true indicates that the channel is designated to be segmentable, and when equal to false indicates that the channel is designated to be non-segmentable.
H223AL1MParameters: is used to indicate parameters specific to using adaptation Layer AL1M.
transferMode indicates whether framed mode or unframed mode is used.
headerFEC indicates whether FEC is SEBCH(16,7) or Golay(24,12).
The length of CRC bits for the payload is indicted by crcLength as 4, 12, 20 or 28 bits.
rcpcCodeRate indicates the RCPC code rate as 8/8, 8/9, ..., 8/32.

arqType indicates the ARQ mode of operation: noARQ indicates no retransmission, typeIArq indicates ARQ type I, and typeIIArq indicates ARQ type II.
alpduInterleaving, if true, indicates the use of AL-PDU interleaving.
alsduSplitting, if true, indicates the use of AL-SDU splitting mode.
H223AL2MParameters: is used to indicate parameters specific to using adaptation Layer AL2M.
headerFEC indicates whether FEC is SEBCH(16,5) or Golay(24,12).
alpduInterleaving, if true, indicates the use of AL-PDU interleaving.
H223AL3MParameters: is used to indicate parameters specific to using adaptation Layer AL3M.
This has the same parameters as AL1MParameters, except transferMode and alsduSplitting are not present.

H223AnnexCArqParameters
numberOfRetransmissions indicates the maximum number of retransmissions that may be used: finite indicates a finite limit on the number of retransmissions that may be used in the range 0 to 16; and infinite indicates that there is no limit on the number of retransmissions that may be used; numberOfRetransmissions equal to the finite value of 0 indicates that the control field is used for the splitting mode but retransmissions are not used.
sendBufferSize indicates the size of the send buffer that will be used, the size being measured in octets.
V76LogicalChannelParameters: is used to indicate parameters specific to using V.76.
audioHeader is used to indicate the use of an audio header on the logical channel. This is a valid parameter for channels of the DataType audio.
suspendResume is used to indicate that the channel may use the suspend/resume procedures to suspend other logical channels. Three channel options may be selected; no suspend resume on the channel, suspend resume using an address or suspend resume without an address as defined in Recommendation V.76. suspendResumewAddress indicates that the suspend/resume channel shall use the address field as defined in Recommendation V.76. suspendResumewoAddress indicates that the suspend/resume channel shall not use the address field.
eRM indicates that the logical channel shall perform error recovery procedures as defined in Recommendation V.76.
uNERM indicates that the logical channel shall operate in non-error recovery mode as defined in Recommendation V.76.
For description of n401, windowSize and loopbackTestProcedure see 12.2.1/V.42, and its subclauses. For the purposes of V.70, n401 shall be encoded in octets.
crcLength is an optional parameter that indicates the CRC length used in error recovery mode. If this parameter is not present, the default CRC length shall be used. crc8bit indicates to use an 8-bit CRC, crc16bit indicates use of the 16-bit CRC and crc32bit indicates to use a 32-bit CRC as defined in Recommendation V.76.
recovery is an optional parameter that indicates the error recover procedures defined in Recommendation V.76. If this parameter is not present, the default error recovery procedure shall be used. sREJ indicates to use the selective frame reject procedure and

mSREJ indicates to use the multiple selective reject procedure as defined in Recommendation V.76.

uIH indicates the use of V.76 UIH frames.

rej indicates the use of the reject procedure in V.76.

V75Parameters is used to indicate parameter specific to using V.75. audioHeaderPresent indicates the presence of the V.75 audio header.

H2250LogicalChannelParameters: is used to indicate parameters specific to using H.225.0. It shall be present in forwardLogicalChannelParameters and reverseLogicalChannelParameters.

The sessionID is a unique RTP or T.120 Session Identifier in the conference. It is used by the transmitter to refer to the session to which the logical channel applies. Only the master can create the session identification. By convention, there are three primary sessions. The first primary session with a session identification of 1 is the audio session, the second primary session with a session identification of 2 is the video session, and the third primary session with a session identification of 3 is the data session. A slave entity can open an additional session by providing a session identification of 0 in the openLogicalChannel message. The master will create a unique session identification and provide it in the openLogicalChannelAck message.

The associatedSessionID is used to associate one session with another. Typical use will be to associate an audio session with a video session to indicate which sessions to process for lip synchronization.

The mediaChannel indicates a transportAddress to be used for the logical channel. It is not present in the OpenLogicalChannel message when the transport is unicast. If the transportAddress is multicast, the master is responsible for creating the multicast transport address and shall include the address in the OpenLogicalChannel message. A slave entity that wishes to open a new multicast channel will provide zeroes in the multicast transportAddress field. The master will create and provide the multicast transportAddress in the OpenLogicalChannelAck message for the slave entity. Note that the MC will use the communicationModeCommand to specify the details about all the RTP Sessions in the conference.

The mediaChannel is used to describe the transport address for the logical channel. IPv4 and IPv6 addresses shall be encoded with the most significant octet of the address being the first octet in the respective OCTET STRING, e.g. the class B IPv4 address 130.1.2.97 shall have the "130" being encoded in the first octet of the OCTET STRING, followed by the "1" and so forth. The IPv6 address a148:2:3:4:a:b:c:d shall have the "a1" encoded in the first octet, "48" in the second, "00" in the third, "02" in the fourth and so forth. IPX addresses, node, netnum, and port shall be encoded with the most significant octet of each field being the first octet in the respective OCTET STRING.

mediaGuaranteedDelivery indicates whether or not the underlying media transport should be selected to provide or not provide guaranteed delivery of data.

mediaControlChannel indicates the media control channel in which the sender of the open logical channel will be listening for media control messages for this session. This field is present only when a media control channel is required.

mediaControlGuaranteedDelivery indicates whether or not the underlying media control transport should be selected to provide or not provide guaranteed delivery of data. This field is present only when a media control channel is required.

The silenceSuppression is used to indicate whether the transmitter stops sending packets during times of silence. It shall be included in the openLogicalChannel message for an audio channel and omitted for any other type of channel.

destination indicates the terminalLabel of the destination if one has been assigned.

dynamicRTPPayloadType indicates a dynamic payload value which is used in H.323 for the H.225.0 alternative H.261 video packetization scheme. This field is present only when a dynamic RTP payload is in use.

mediaPacketization indicates which optional media packetization scheme is in use.

redundancyEncoding indicates that the redundant encoding method indicated in this parameter is to be used for the logical channel to be opened. The primary encoding is defined by the *dataType* of the *forwardLogicalChannelParameters* or the *reverseLogicalChannelParameters*, respectively. The type of redundancy encoding to be applied for this logical channel is identified by the *redundancyEncodingMethod* parameter, the secondary encoding is specified in the *secondaryEncoding* parameter. The *DataType* (audio, video, etc.) selected for both primary and secondary encoding shall match and shall be in accordance with the *redundancyEncodingMethod* selected.

The source parameter is used to identify the terminal number of the sender of the OpenLogicalChannel message.

h235 Key: is used to include, and specify the method by which media specific session keys are protected as they are passed between two endpoints. The encoding of this field is a nested ASN.1 value as described in Recommendation H.235.

EscrowData is used to specify the type and contents of any key escrow mechanism in use. Specific types and contents may be required by implementations when media encryption is enabled.

T120SetupProcedure indicates how the T.120 conference is to be set up. For originateCall and waitForCall, the caller should derive the T.120 numeric conference name from the H.323 CID (as described in Recommendation H.323), and issue the appropriate PDU (if the endpoint is master, it should issue an invite request, while a slave should issue a join request). For issueQuery, the caller should first issue a query request, and then set up the T.120 conference in accordance with the contents of the query response (as described in Recommendation T.124).

SYSTEM DESIGN

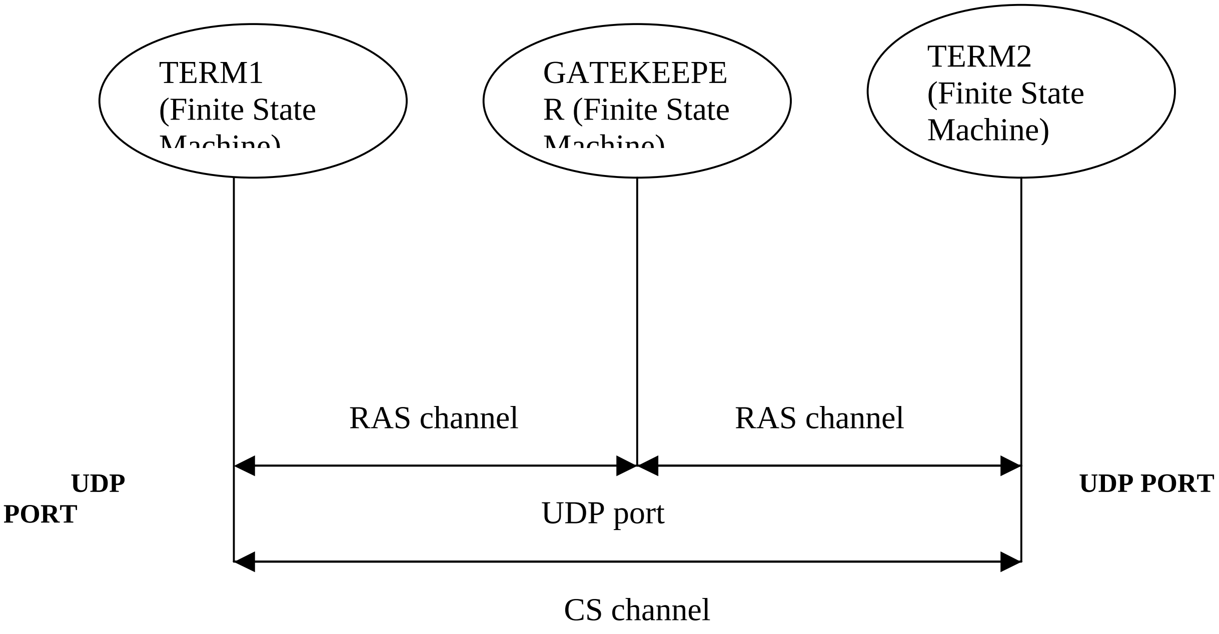
TERM1 (Finite State Machine)
GATEKEEPER (Finite State Machine)
TERM2 (Finite State Machine)
RAS channel
RAS channel
UDP PORT
UDP PORT
UDP port
CS channel

Figure 2: System Design

The system consists of three finite state machines:

- Terminal 1
- Terminal 2
- Gatekeeper

All the finite state machines have unique initial state and a unique final state. The state changes upon receiving or sending messages on their UNIX System V /

LINUX sockets (end point for communications). There are two kinds of sockets used by each terminal.

The first one is a Datagram socket, which uses a UDP (connectionless) and communicates with the gatekeeper on it. The second one is a Stream socket which uses a TCP (connection oriented) and communicates with another Stream socket created by another terminal end point.

SOCKETS

NAME

socket - create an endpoint for communication

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

DESCRIPTION

Socket creates an endpoint for communication and returns a descriptor. The domain parameter specifies a communications domain within which communication will take place; this selects the protocol family, which should be used. These families are defined in the include file sys/socket.h. The currently understood formats are

AF_UNIX (UNIX internal protocols)

AF_INET (ARPA Internet protocols)

AF_ISO   (ISO protocols)

AF_NS   (Xerox Network Systems protocols)

AF_IMPLINK
(IMP "host at IMP" link layer)

The socket has the indicated type, which specifies the semantics of communication. Currently defined types are:

SOCK_STREAM
SOCK_DGRAM

SOCK_RAW
SOCK_SEQPACKET
SOCK_RDM

A SOCK_STREAM type provides sequenced, reliable, two-way connection based byte streams. An out-of-band data transmission mechanism may be supported. A SOCK_DGRAM socket supports datagrams (connectionless, unreliable messages of a fixed(typically small) maximum length). A SOCK_SEQPACKET socket may provide a Sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer may be required to read an entire packet with each read system call. This facility is protocol specific, and presently implemented only for AF_NS. SOCK_RAW sockets provide access to internal network protocols and interfaces.

The protocol specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family. However, it is possible that many protocols may exist, in which case a particular protocol must be specified in this manner. The protocol number to use is particular to the communication domain" in which communication is to take place. Sockets of type SOCK_STREAM are full-duplex byte streams, similar to pipes. A stream socket must be in a connected state before any data may be sent or received on it. A connection to another socket is created with a connect() call. Once connected, data may be transferred using read() and write() calls or some variant of the send() and recv() calls. When a session has been completed a close() may be performed. Out-of-band data may also be transmitted as described in send() and received as described in recv().The communications protocols used to implement a SOCK_STREAM insure that data is not lost or duplicated.

If a piece of data for which the peer protocol has buffer space cannot be successfully ransmitted within a reasonable length of time,then the connection is considered broken and calls will indicate an error with -1 returns and with ETIMEDOUT as the specific code in the global variable errno. The protocols optionally keep sockets warm by forcing transmissions roughly every minute in the absence of other activity. An error is then indicated if no response can be elicited on an otherwise idle connection for a extended period (e.g. 5 minutes). A SIGPIPE signal is raised if a process sends on a broken stream; this causes naive processes, which do not handle the signal, to exit. SOCK_SEQPACKET sockets employ the same system calls as SOCK_STREAM sockets. The only difference is that read() calls will return only the amount of data requested, and any remaining in the arriving packet will be discarded. SOCK_DGRAM and SOCK_RAW sockets allow sending of datagrams to correspondents named in send() calls. Datagrams are generally received with recvfrom(), which returns the next datagram with its return address.

An fcntl() call can be used to specify a process group to receive a SIGURG signal when the out-of-band data arrives. It may also enable non-blocking I/O and asynchronous notification of I/O events via SIGIO. The operation of sockets is controlled by socket level options. These options are defined in the file sys/socket.h. Setsockopt() and getsockopt() are used to set and get options, respectively.

RETURN VALUES

A -1 is returned if an error occurs, otherwise the return value is a descriptor referencing the socket.

NAME

bind - bind a name to a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int  bind(int  sockfd, struct sockaddr *my_addr, socklen_t
addrlen);
```

DESCRIPTION

bind gives the socket, sockfd, the local address my_addr. my_addr is addrlen bytes long. Traditionally, this is called "assigning a name to a socket" (when a socket is created with socket(), it exists in a name space (address family) but has no name assigned.)

NOTES

Binding a name in the UNIX domain creates a socket in the file system that must be deleted by the caller when it is no longer needed (using unlink()). The rules used in name binding vary between communication domains. Consult the manual entries in section 4 for
detailed information.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

NAME

send, sendto, sendmsg - send a message from a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>


int send(int s, const void *msg, size_t len, int flags);

int  sendto(int s, const void *msg, size_t len, int flags,
const struct sockaddr *to, socklen_t tolen);

int sendmsg(int s, const struct msghdr *msg, int flags);
```

DESCRIPTION

Send, sendto, and sendmsg are used to transmit a message to another socket. Send may be used only when the socket is in a connected state, while sendto and sendmsg may be used at any time.

The address of the target is given by to with tolen specifying its size. The length of the message is given by len. If the messages too long to pass atomically through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted.

No indication of failure to deliver is implicit in a send. Locally detected errors are indicated by a return value of -1. If no messages space is available at the socket to hold

the message to be transmitted, then send normally blocks, unless the socket has been placed in non-blocking I/O mode. The select() call may be used to determine when it

is possible to send more data.

The flags parameter may include one or more of the following:

```
 #define   MSG_OOB         0x1  /* process out-of-band data */
 #define   MSG_DONTROUTE   0x4  /* bypass routing, use direct interface
*/
```

The flag MSG_OOB is used to send out-of-band data on sockets that support this notion (e.g.SOCK_STREAM); the underlying protocol must also support out-of-band data.

MSG_DONTROUTE is usually used only by diagnostic or routing programs.

RETURN VALUES

The call returns the number of characters sent, or -1 if an error occurred.

NAME

recv, recvfrom, recvmsg - receive a message from a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>


int recv(int s, void *buf, int len, unsigned int flags);

int recvfrom(int s, void *buf, int len, unsigned int flags
struct sockaddr *from, socklen_t *fromlen);

int  recvmsg(int      s,  struct  msghdr  *msg,  unsigned  int
flags);
```

DESCRIPTION

The recvfrom and recvmsg are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.

If from is non-nil, and the socket is not connection-oriented, the source address of the message is filled in. Fromlen is a value-result parameter, initialized to the size of the buffer associated with from, and modified on return to indicate the actual size of the address stored there.

The recv call is normally used only on a connected socket and is identical to recvfrom with a nil from parameter.As it is redundant, it may not be supported in future releases.

All three routines return the length of the message on successful completion. If a message is too long to fit in the supplied buffer, excess bytes may be discarded depending on the type of socket the message is received from.

If no messages are available at the socket, the receive call waits for a message to arrive, unless the socket is nonblocking in which case the value -1 is returned and the

external variable errno set to EWOULDBLOCK.The receive calls normally return any data available, up to the requested amount, rather than waiting for receipt if the full amount requested; this behavior is affected by the socket-level options SO_RCVLOWAT and

SO_RCVTIMEO described in getsockopt().The select() call may be used to determine when more data arrive.

The flags argument to a recv call is formed by or'ing one or more of the values:

MSG_OOB process out-of-band data

MSG_PEEK
peek at incoming message

MSG_WAITALL
wait for full request or error

The MSG_OOB flag requests receipt of out-of-band data that would not be received in the normal data stream. Some protocols place expedited data at the head of the normal data queue, and thus this flag cannot be used with such protocols. The MSG_PEEK flag causes the receive operation to return data from the beginning of the receive queue without removing that data from the queue.Thus, a subsequent receive call will return the
same data. The MSG_WAITALL flag requests that the operation block until the full request is satisfied. However, the call may still return less data than requested if a signal is caught, an error or disconnect occurs, or the next data to be received is of a different type than that returned.

RETURN VALUES
These calls return the number of bytes received or -1 if an error occurred.

NAME
connect - initiate a connection on a socket

SYNOPSIS
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen);

DESCRIPTION
The parameter sockfd is a socket. If it is of type SOCK_DGRAM, this call specifies the peer with which the socket is to be associated; this address is that to which datagrams are to be sent, and the only address from which datagrams are to be received. If the socket is of type SOCK_STREAM, this call attempts to make a connection to another socket. The other socket is specified by serv_addr, which is an address in the communications space
of the socket. Each communications space interprets the serv_addr, parameter in its own way. Generally, stream sockets may successfully connect only once; datagram sockets may use connect multiple times to change their association. Datagram sockets may dissolve the association by connecting to an invalid address, such as a null address.

RETURN VALUE

If the connection or binding succeeds, zero is returned. On error, -1 is returned, and errno is set appropriately.

NAME

listen - listen for connections on a socket

SYNOPSIS

```
#include <sys/socket.h>

int listen(int s, int backlog);
```

DESCRIPTION

To accept connections, a socket is first created with socket(), a willingness to accept incoming connections and a queue limit for incoming connections are specified with listen, and then the connections are accepted with accept(). The listen call applies only to sockets of type SOCK_STREAM or SOCK_SEQPACKET. The backlog parameter defines the maximum length the queue of pending connections may grow to. If a connection request arrives with the queue full the client may receive an error with an indication of ECONNREFUSED, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

NAME

accept - accept a connection on a socket

SYNOPSIS

```
#include <sys/types.h>

#include <sys/socket.h>

int   accept(int       s,  struct  sockaddr    *addr,  socklen_t *addrlen);
```

DESCRIPTION

The argument s is a socket that has been created with socket(), bound to an address with bind(), and is listening for connections after a listen(). The accept function extracts the first connection request on the queue of pending connections, creates anew socket with the same properties of s and allocates a new file descriptor for the socket. If no pending connections are present on the queue,

and the socket is not marked as non-blocking, accept blocks the caller until a connection is present. If the socket is marked non-blocking and no pending connections are present on the queue, accept returns an
error as described below. The accepted socket may not be used to accept more connections. The original sockets remains open.
The argument addr is a result parameter that is filled in with the address of the connecting entity, as known to the communications layer. The domain in which the communication is occurring determines the exact format of the addr parameter. The addrlen is a value-result parameter; it should initially contain the amount of space
Pointed to by addr; on return it will contain the actual length (in bytes) of the address returned. This call is used with connection-based socket types, currently with
SOCK_STREAM.

It is possible to select() a socket for the purposes of doing an accept by selecting it for read.
RETURN VALUES
The call returns -1 on error. If it succeeds, it returns a non-negative integer that is a descriptor for the accepted socket.

NAME
send, sendto, sendmsg - send a message from a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>


int send(int s, const void *msg, size_t len, int flags);

int  sendto(int s, const void *msg, size_t len, int flags,
const struct sockaddr *to, socklen_t tolen);

int sendmsg(int s, const struct msghdr *msg, int flags);
```

DESCRIPTION
Send, sendto, and sendmsg are used to transmit a message to another socket. Send may be used only when the socket is in a connected state, while sendto and sendmsg may be used at any time.
The address of the target is given by to with tolen specifying its size. The length of the message is given by len. If the message is too long to pass atomically

through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted.

No indication of failure to deliver is implicit in a send. Locally detected errors are indicated by a return value of -1.

If no messages space is available at the socket to hold the message to be transmitted, then send normally blocks, unless the socket has been placed in non-blocking I/O

mode. The select() call may be used to determine when it is possible to send more data.

The flag parameter may include one or more of the following:

```
     #define     MSG_OOB          0x1  /* process out-of-band data */
     #define     MSG_DONTROUTE   0x4   /* bypass routing, use direct interface
*/
```

RETURN VALUES

The call returns the number of characters sent, or -1 if an error occurred.

NAME

recv, recvfrom, recvmsg - receive a message from a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>


int recv(int s, void *buf, int len, unsigned int flags);

int recvfrom(int s, void *buf, int len, unsigned int flags
struct sockaddr *from, socklen_t *fromlen);

int recvmsg(int s, struct msghdr *msg, unsigned int
flags);
```

DESCRIPTION

The recvfrom and recvmsg are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.If from is non-nil, and the socket is not connection-oriented, the source address of the message is filled in.

Fromlen is a value-result parameter, initialized to the size of the buffer associated with from, and modified on return to indicate the actual size of the address stored there.

The recv call is normally used only on a connected socket and is identical to recvfrom with a nil from parameter.

TERMINALS ON RAS PORT (UDP)

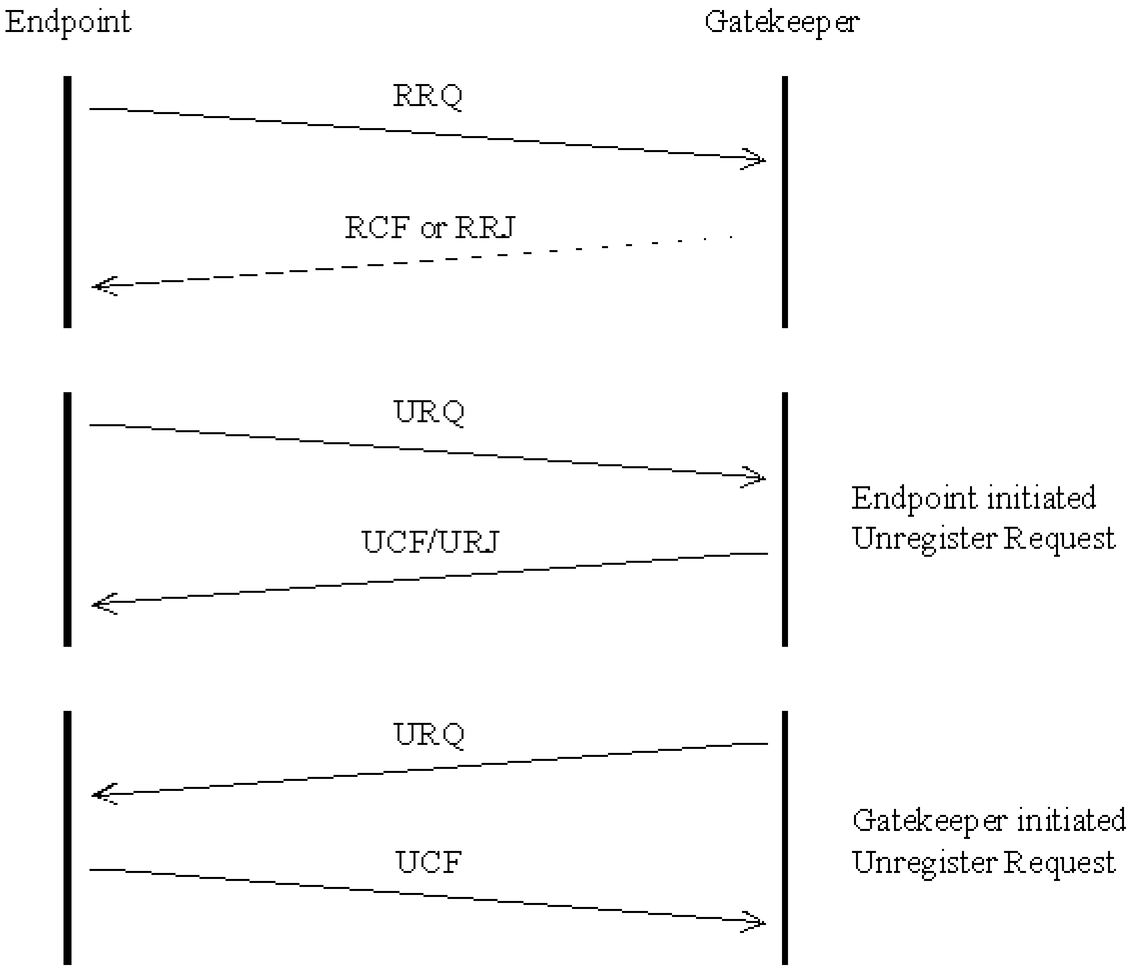

Figure 3 Terminals on RAS

The terminals are implemented as a finite state machine and they create two sockets:

The first one is the datagram socket (SOCK_DGRAM) through which it communicates with the datagram socket of the gatekeeper. It passes its transport address and the port numbers of both the sockets and the gatekeeper passes on these address to the second terminal to which the first terminal wants to communicate and vice versa. The implementation of the terminals as finite sate machines during the admission, registration and status cycles are given below (refer to APPENDIX):

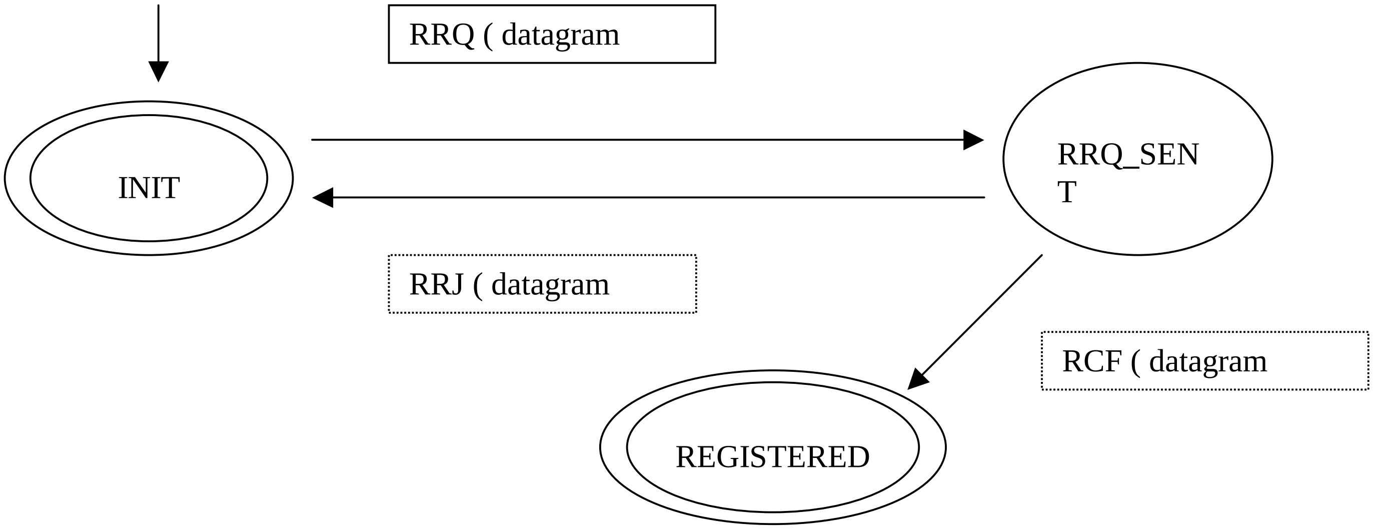

Figure 4: Registration of terminal with the Gatekeeper

The terminal has the capability of locating query i.e. it can give an alias name to the gatekeeper to check whether a terminal is registered with that alias name. If the gatekeeper is to locate the alias in its registration record then it sends a LCF to the terminal else it sends a LCF. The process is given below in the form of a state diagram (refer to APPENDIX):

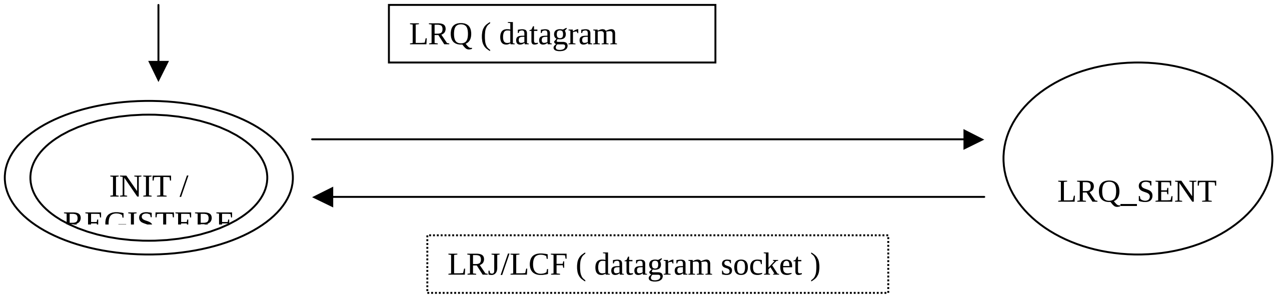

Figure 5: Locate Query done by the terminal

The terminal can also change its bandwidth when it is communicating with another terminal on a control signal or a TCP port/stream socket. The process is given below as a state diagram (refer to APPENDIX):

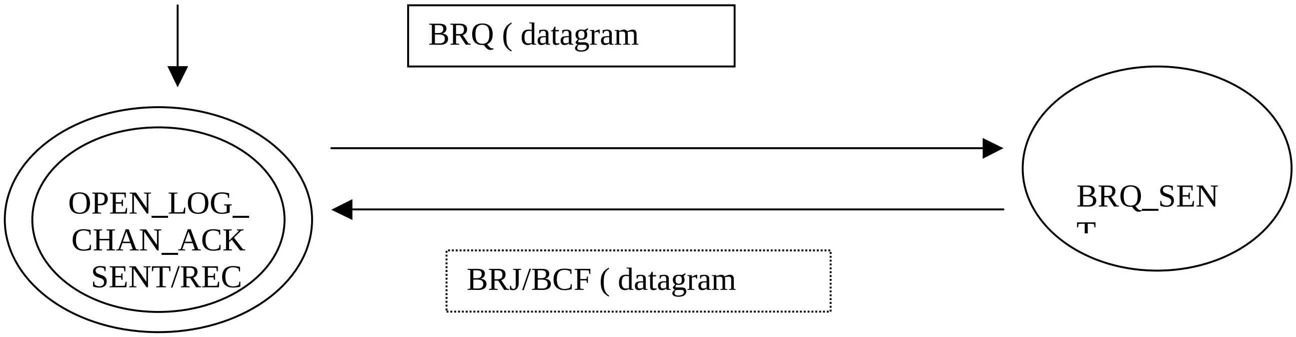

Figure 6: Change of Bandwidth request by a terminal

A registered terminal has to unregister with the gatekeeper before it can exit from the network. The process of unregistering is given as a state diagram below (refer to APPENDIX):

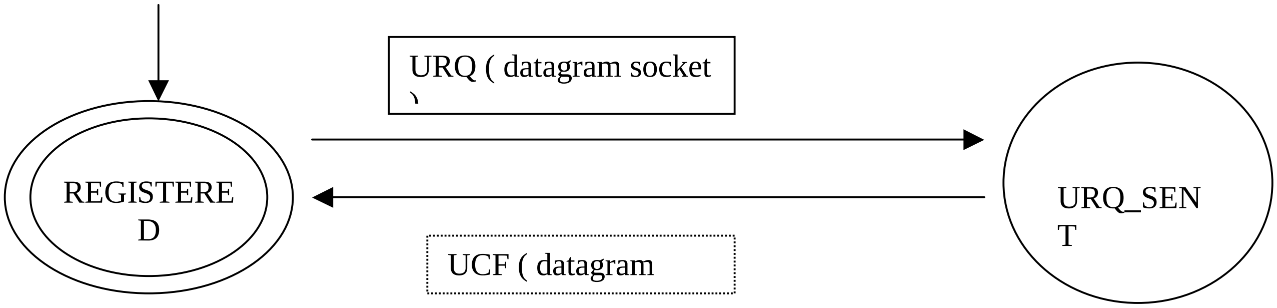

Figure 7: Unregistration Cycle of the terminal

GATEKEEPER ON RAS/UDP PORT

The gatekeeper is implemented just like a terminal i.e. it is also a finite state machine. It creates a datagram socket (SOCK_DGRAM). The gatekeeper then binds the socket to a transport address (IP address + port No.) which is known to every terminal who wants to register with the gatekeeper in the network. The gatekeeper sends confirmation/reject messages to the terminal, which sends requests like:

- Registration,
- Admission,
- Bandwidth change,
- Locate Query,
- Disengage and
- Unregister.

The gatekeeper always communicates with the terminals on its datagram socket.

The entire process has been illustrated below (refer to APPENDIX):

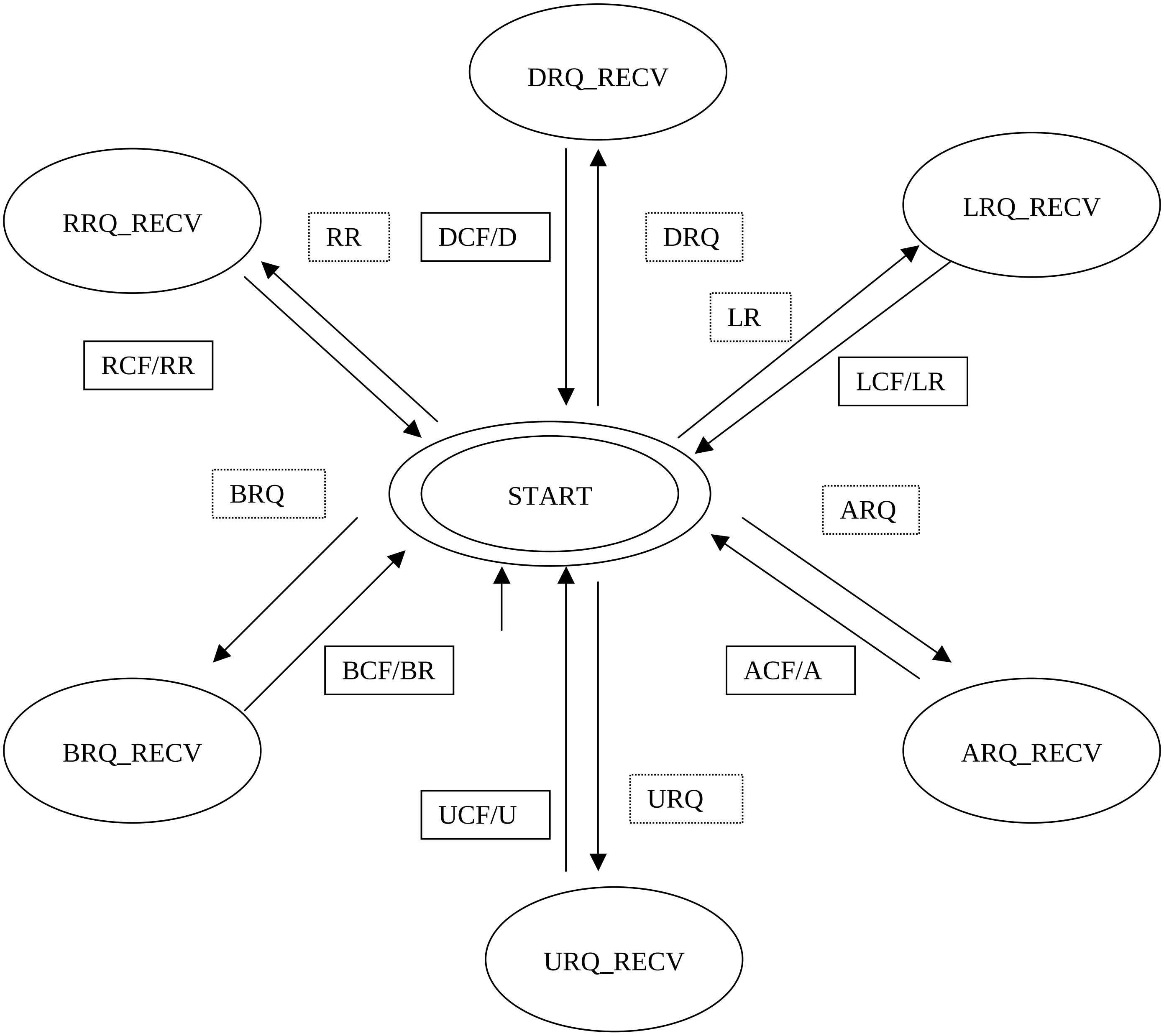

Figure 8: Registration, Admission, Bandwidth management and Status control by the gatekeeper

CALL SIGNALING/CALL SETUP

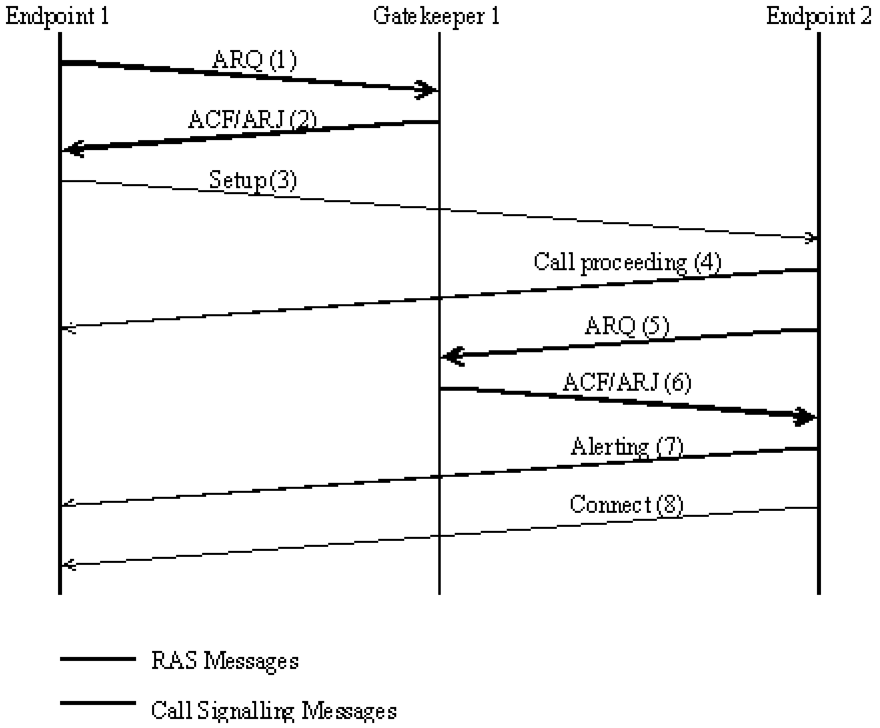

Figure 9: Call Setup for registered terminals

In the scenario shown in Figure 9, both endpoints are registered to the same Gatekeeper, and the Gatekeeper has chosen Direct Call Signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with that Gatekeeper. The Gatekeeper shall return the Call Signalling Channel Transport Address of Endpoint 2 (called endpoint) in the ACF. Endpoint 1 then sends the Setup (3) message to Endpoint 2 using that Transport Address. If Endpoint 2 wishes to accept the call, it initiates an ARQ (5)/ACF (6) exchange with the Gatekeeper. It is possible that an ARJ (6) is received by Endpoint 2, in which case it sends Release Complete to Endpoint 1. Endpoint 2 responds with the

Connect (8) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

This mechanism is implemented using the following finite state machines for the terminals and the state diagram for the gatekeeper has already been dealt with in Fig. 8.

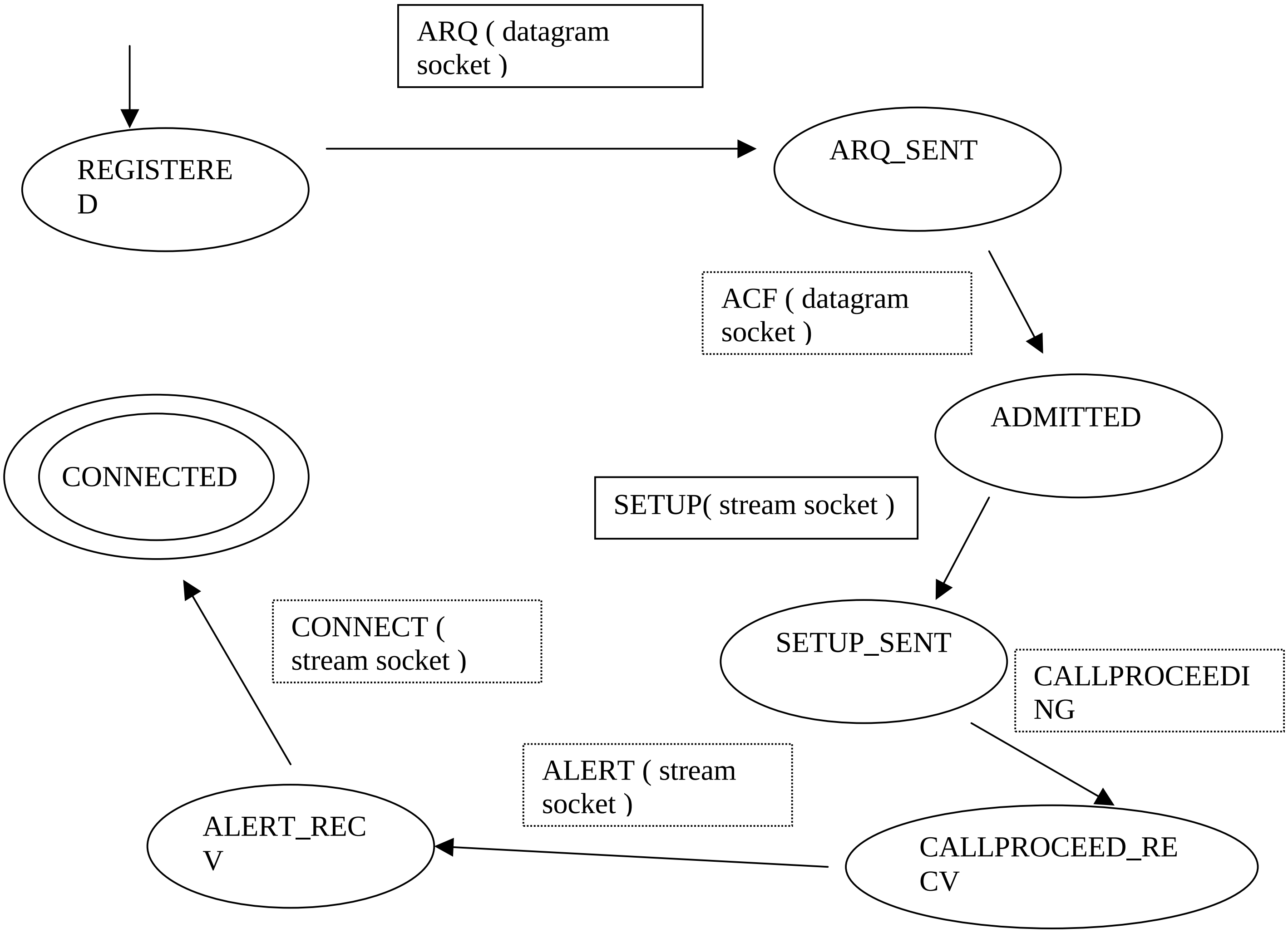

Figure 10: Call Setup for terminal 1 (calling party)

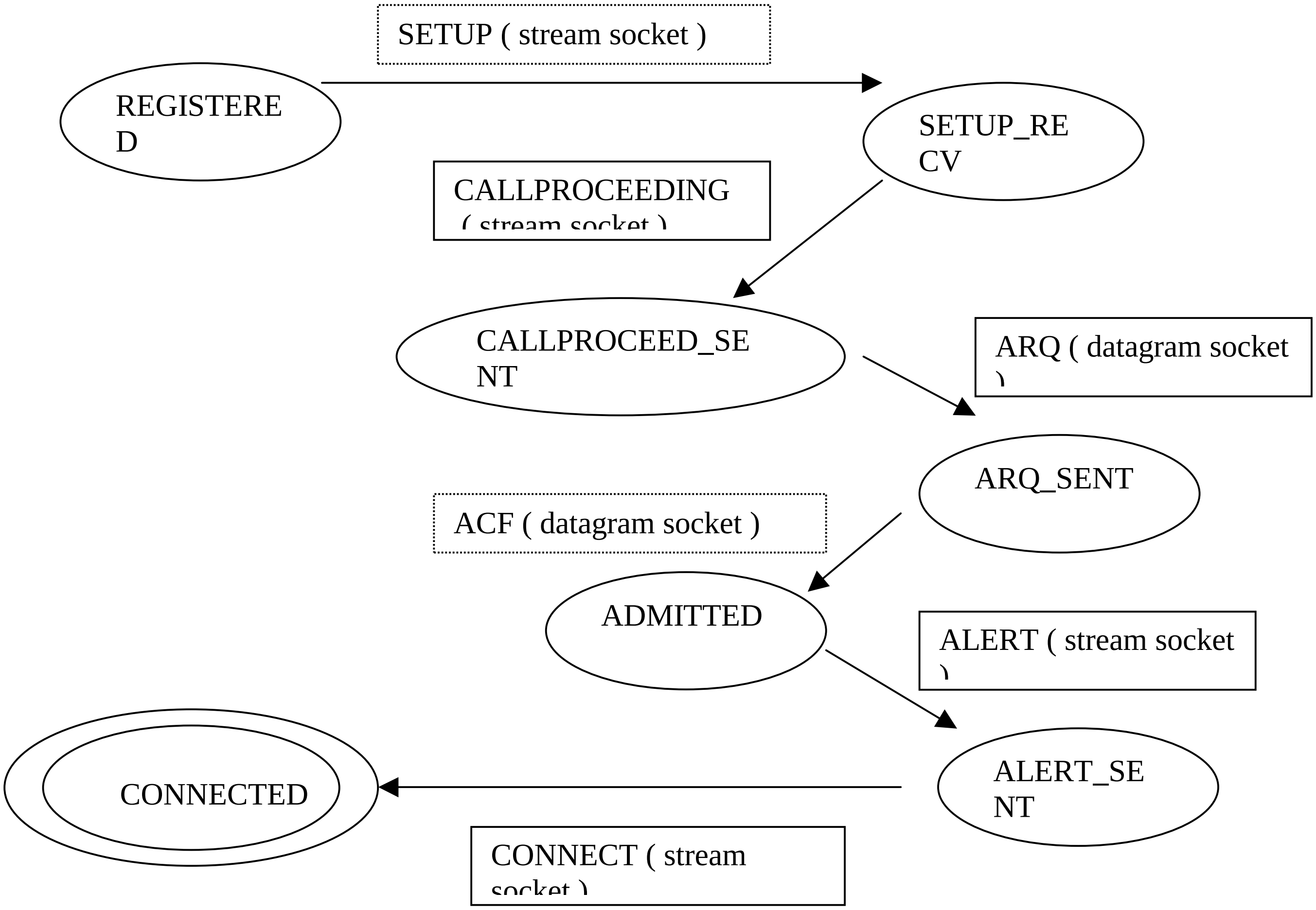

Figure 11: Call Setup for terminal 2 (called party)

CONTROL SIGNALING

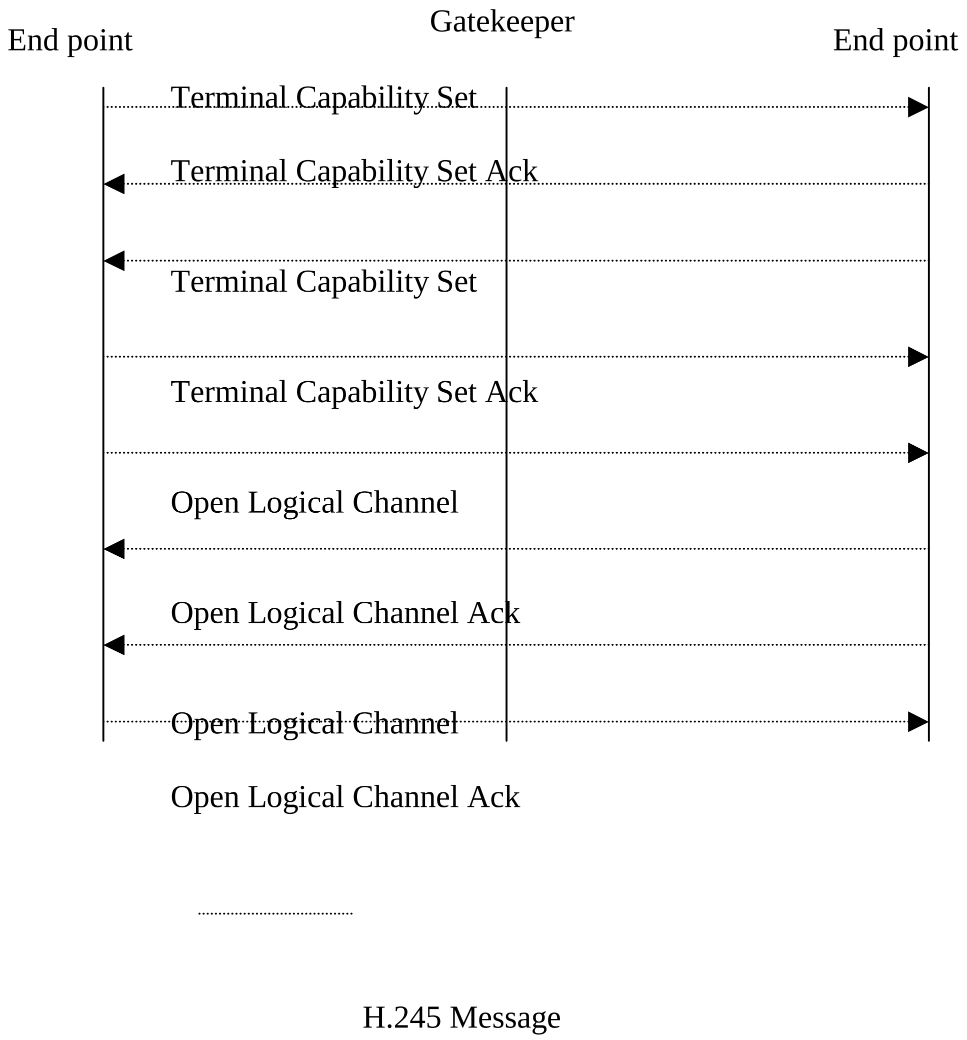

Figure 12: Control Signaling flows

Control Signaling is a mechanism used for creation of logical media channels for transfer of data. The first step is to exchange the terminal capabilities of one another. The next step is the creation of the logical channel which includes the transport address of the channel which is actually used for the transfer of media streams or the data.

This has also been implemented as a finite state machine and the flow of data can occur only after the control signaling has been implemented.

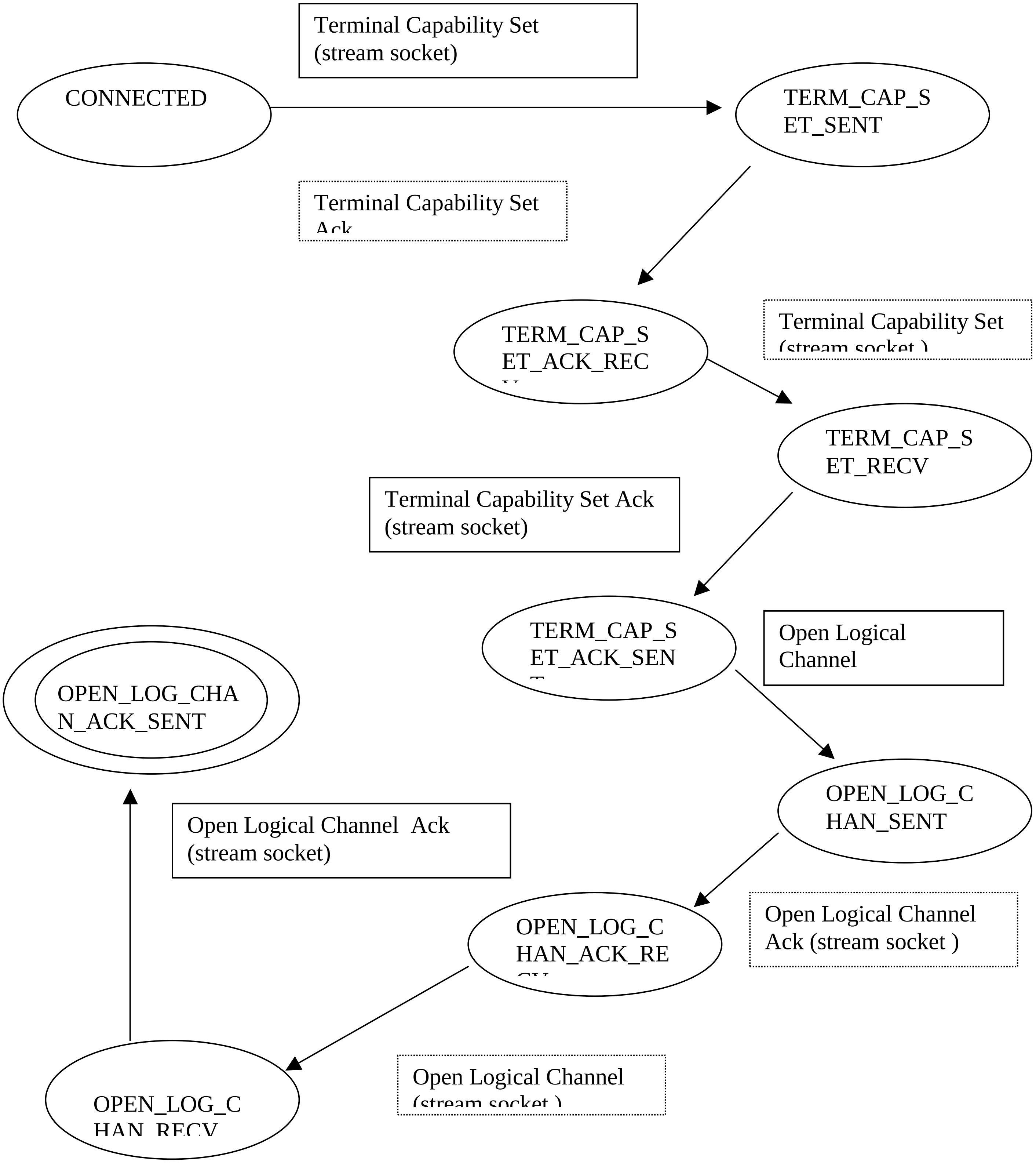
Terminal Capability Set
(stream socket)
CONNECTED
TERM_CAP_S
ET_SENT
Terminal Capability Set
Ack
TERM_CAP_S
ET_ACK_REC
V
Terminal Capability Set
(stream socket)
TERM_CAP_S
ET_RECV
Terminal Capability Set Ack
(stream socket)
TERM_CAP_S
ET_ACK_SEN
T
Open Logical
Channel
OPEN_LOG_CHA
N_ACK_SENT
OPEN_LOG_C
HAN_SENT
Open Logical Channel Ack
(stream socket)
OPEN_LOG_C
HAN_ACK_RE
CV
Open Logical Channel
Ack (stream socket)
OPEN_LOG_C
HAN_RECV
Open Logical Channel
(stream socket)

Figure 13: Control Signaling flows for both the terminals

CALL RELEASE

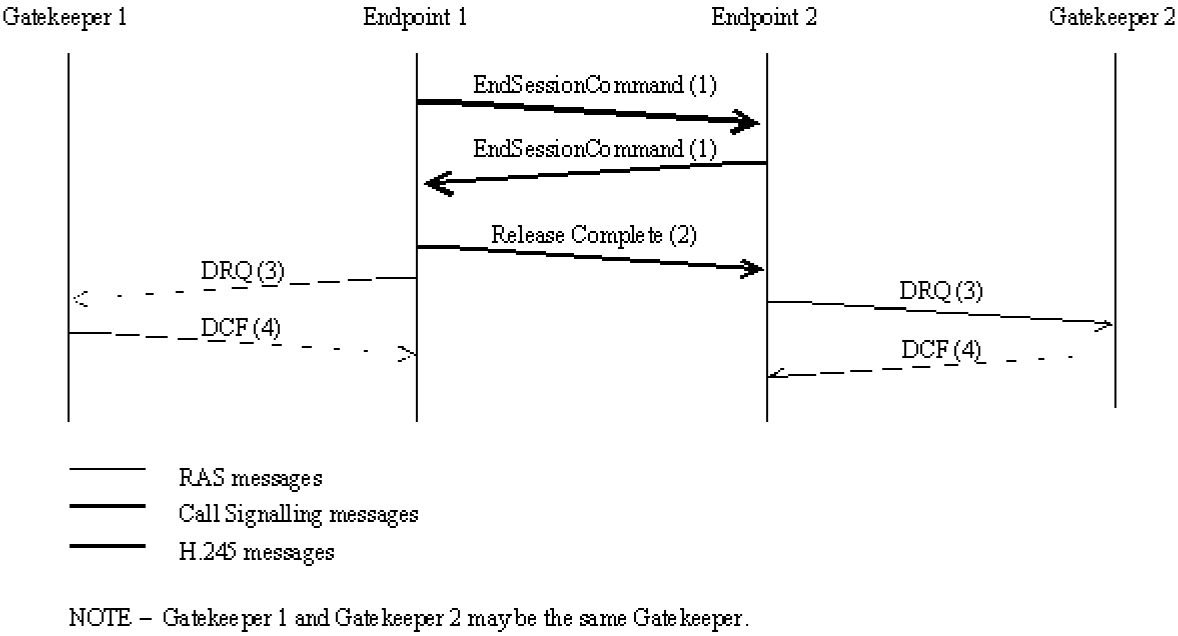

Figure 14: Call Termination

In networks that contain a Gatekeeper, the Gatekeeper needs to know about the release of bandwidth. After performing steps 1) to 6) above, each endpoint shall transmit an H.225.0 Disengage Request (DRQ) message (3) to its Gatekeeper. The Gatekeeper shall respond with a Disengage Confirm (DCF) message (4). After sending the DRQ message, the endpoints shall not send further unsolicited IRR messages to the Gatekeeper. See Figure 14. At this point, the call is terminated. Figure 14 shows the direct call model; a similar procedure is followed for the Gatekeeper routed model.
The DRQ and DCF messages shall be sent on the RAS Channel.

We have implemented the call termination sequence for both the terminals with only one gatekeeper in the network (gatekeeper 1 and gatekeeper 2 being identical in the figure).

The state transitions have been shown below:

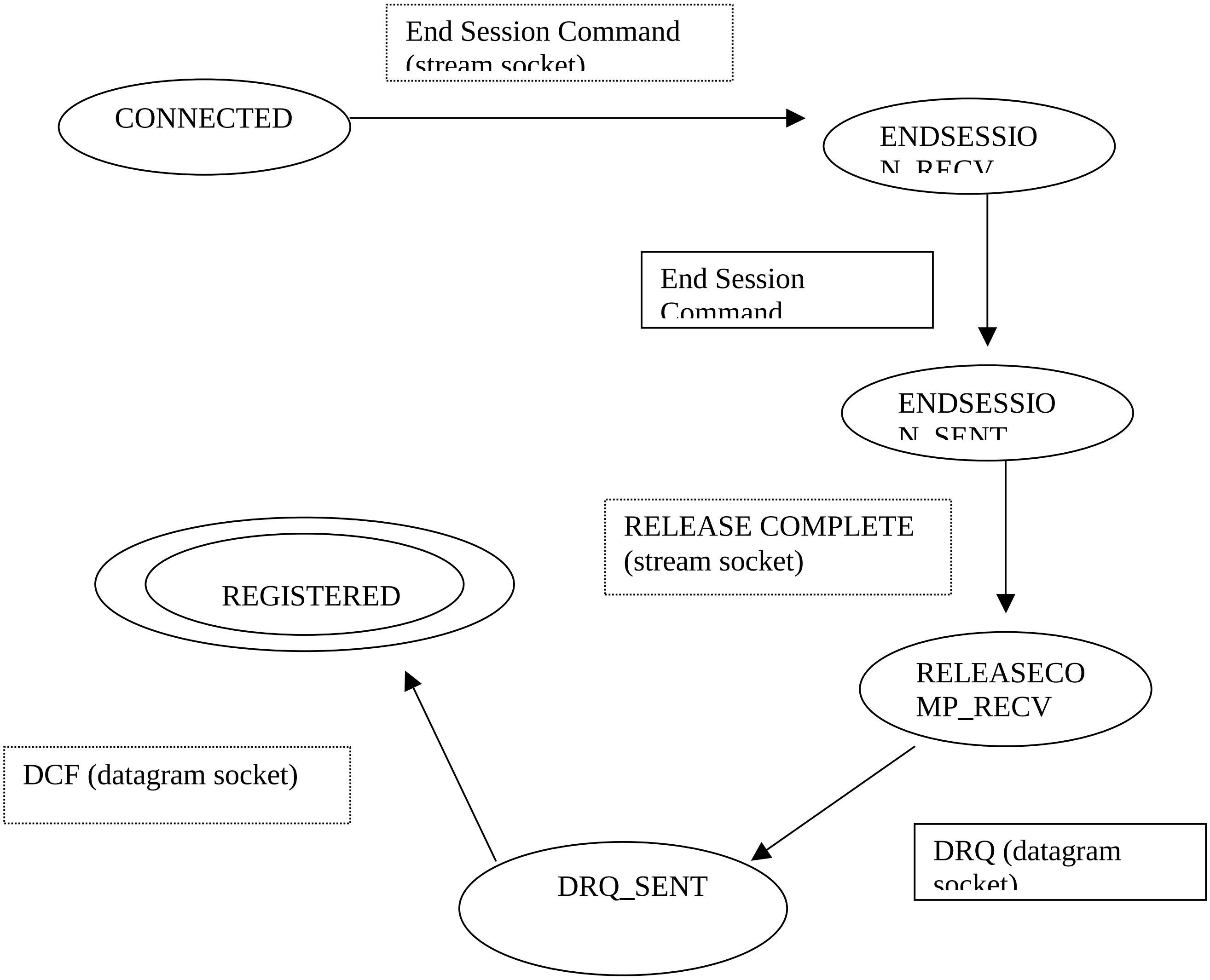

Figure 15: Call Release

SCOPE OF FUTURE WORK

The above attempt though implements the H.323 standard successfully, yet there can be many key areas in which a lot of extensions can be done. As already mentioned before the H.323 components include terminals, gatekeepers, gateways and multipoint control units (MCUs), correspondingly each implementation done in this project can be supplemented
by adding many new features. This discussion shall present the key areas for future work.

Adding features like RTP/RTCP for sequencing audio/video packets can extend the terminal characteristics. H.323 terminals may also support the G.711 audio codec. Video codecs can also be included in the terminals.

The terminal capabilities can be increased to their fullest extent by adding features like data-conferencing and MCU capabilities thus extending the idea of point to point connection to multipoint connections as used in the case of online audio/video conferencing over IP.

The gatekeeper characteristics may be extended by including services like:

- Gatekeeper routed call signaling
- Accounting
- Billing
- Call Authorization
- Call Management

The Gatekeeper can be used to route all the messages (call signaling and control signaling), which earlier were exchanged directly between the terminals. For this purpose, the gatekeeper creates a TCP stream socket and routes the incoming messages to the destination. When an end point sends call signaling messages to the gatekeeper, the gatekeeper may accept or reject the call according to H.225 specifications. The reasons for rejection may include access based or time based restrictions, to and from particular terminals or gateways. The gatekeeper may maintain information about all active calls so that it can control its zone by providing the maintained information to the bandwidth management function or by rerouting the calls to different end points to achieve load balancing.

APPENDIX

ABBREVIATIONS

ACF	Admission Confirmation
ARJ	Admission Reject
ARQ	Admission Request
ATC	ATM Transfer Capability
ATM	Asynchronous Transfer Mode
BCF	Bandwidth Change Confirmation
B-ISDN	Broadband Integrated Services Digital Network
BRJ	Bandwidth Change Reject
BRQ	Bandwidth Change Request
DCF	Disengage Confirmation
DRQ	Disengage Request
GCF	Gatekeeper Confirmation
GK	Gatekeeper
GRJ	Gatekeeper Reject
GRQ	Gatekeeper Request
GW	Gateway
IP	Internet Protocol
IPX	Internetwork Protocol Exchange
ISDN	Integrated Services Digital Network
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LAN	Local Area Network
LCF	Location Confirmation
LRJ	Location Reject
LRQ	Location Request
MCU	Multipoint Control Unit
QOS	Quality of Service
RAS	Registration, Admission and Status
RCF	Registration Confirmation
RRJ	Registration Reject
RRQ	Registration Request
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SCN	Switched Circuit Network
TCP	Transport Control Protocol
UCF	Unregister Confirmation
UDP	User Datagram Protocol
URJ	Unregister Reject
URQ	Unregister Request

LEGENDS

Symbols used in the state diagrams and their meanings are listed below:

FIGURE	MEANING
→ ⬭	INITIAL STATE
⬭	STATE
◎	FINAL STATE
▭	MESSAGE SENT
⬚	MESSAGE RECEIVED
→	STATE TRANSITION

References

1. UNIX NETWORK PROGRAMMING by W.RICHARD STEVENS, Prentice Hall of India PUBLICATIONS 1999 edition (10th reprint)
2. INTRODUCING UNIX SYSTEM V by RACHEL MORGAN and HENRY MCGILTON, McGraw – Hill Publications 1987 edition
3. PROGRAMMING WITH C by BYRON S. GOTTFRIED Tata McGraw – Hill Publications 1997 edition (18th reprint)
4. H.323 TUTORIAL by Web ProForum Tutorials Copyright © The International Engineering Consortium http://www.iec.org
5. ITU-T Recommendation H.225.0
6. ITU-T Recommendation H.245.0
7. ITU-T Recommendation H.323