Telecommunication Standardization Sector
Original: English
(TSS)

AVC-825

Experts Group for Video Coding and Systems in ATM and Other Network Environments

October 23-28, 1995

STUDY GROUP 15
CONTRIBUTION

Source: Vineet Kumar, Intel Corporation
email: vineet_kumar@ccm.jf.intel.com
voice: +1 (503) 264-3439
fax: +1 (503) 264-3375

Daniel A. Brown
email: daniel_a_brown@ccm.jf.intel.com
voice: +1 (503) 264-8610

Title: Conference Control in H.323

Date: September 29, 1995

1. Introduction

This proposal describes Conference Control and how H.245 is extended to include multicast in the H.323/H.22z environment. It does not include H.230 commands needed for Multipoint support. Relevant H.230 command, such as chair control and terminal number assignment, will have to be added to H.245. This proposal assumes that connection setup has been successfully completed and a reliable connection has been established with H.245's logical channel 0 used for control. Capability exchange occurs on logical channel 0. At the end of the capability exchange, other logical channels may be opened depending on the capabilities of the two entities involved.

2. Conference Models

There are essentially three models for conference control. These are:

1. *Centralized.* Control, data, audio, and video are centralized. Here the MC does audio mixing and video switching for all the endpoints involved in the conference.
2. *Audio and Video Distributed.* Control and Data are centralized whereas video and audio are distributed through multicast. Here the endpoints receive all the audio and video streams and do the audio mixing and video switching (and/or continuous presence). The control is, however, centralized but the functionality of the MC can be implemented in the endpoints/gateways. Any one endpoint/gateway can be selected to be the MC for a conference. This model will be referred to as the multicast model in the rest of this proposal.
3. *Distributed.* Control and data are distributed through reliable multicast whereas audio and video are distributed through unreliable multicast. There is no centralized MC. This fully distributed model is beyond the scope of H.323.

3. Conference Types

Conferences can occur in one of three ways within the context of model 1 and model 2:

1. *Point-to-Point conference.* Here one endpoint calls another endpoint. After connection setup and capability exchange, both endpoints operate within the capabilities exchanged.
2. *Predetermined MC based conference.* Here the MC is known to all participants of the conference. Each endpoint calls the MC or the MC invites the endpoints to join the conference.
3. *Ad Hoc conference.* Here an endpoint calls another endpoint and starts a point-to-point conference. If this conference will become multipoint, an MC is selected between the two endpoints. The MC can then invite other endpoints to join the conference. The MC must be available for the entire duration of the conference. But the participant on the same endpoint as the MC can leave the conference at any time.

4. Considerations

4.1 Time-To-Live (TTL)

The TTL field in the IP header should be set to the maximum as a default. A conference that has a good idea about the distances of the participants (router hops) should set its own TTL value to restrict the conference within a certain range.

4.2 Multicast Addresses

Unique multicast addresses within the range (determined by TTL) of the conference must be dynamically assigned by the MC. Multicast addresses could be reserved in advance or picked on-the-fly. The method used by the MC to pick unique multicast addresses is beyond the scope of H.323. Currently, there is no standard way to get unique multicast addresses. One could potentially build a Multicast Server that has a knowledge of multicast addresses in use. The MC could request the Server for unique multicast addresses.

4.3 Transport addresses (port numbers)

For the centralized conference, port numbers can be negotiated between the two entities (two endpoints or an endpoint and a MC) involved.

In the multicast conference, all endpoints must receive a stream on the same port number. Static allocation of port numbers will not work if multiple conferences can be launched from one endpoint. Therefore, the MC must dynamically pick unique port numbers. And endpoints joining the conference must detect collisions and take an appropriate action. The action may involve either closing those applications which are using the port numbers of the conference or to disconnect from the conference.

The alternative to making endpoints detect and resolve collisions is to have the MC change the port numbers every time an endpoint joining the conference reports a collision. This alternative is undesirable for two reasons: (1) Changing port numbers requires closing the existing logical channel and then opening a new one, (2) It gets harder to reach a consensus on port number as the number of endpoints in the conference increase.

Another alternative is a combination of static and dynamic allocation of port numbers. A range of port numbers can be statically assigned to H.323 conferences. And the MC can dynamically assign port numbers from the static range during the conference.

4.4 Processing power available to the endpoint

In a multicast conference, the endpoint will be receiving the audio and video streams of all the endpoints involved in the conference. Due to processing constraints, the endpoint will have to specify the number of audio, video, and data streams that it can handle simultaneously. This can be done through the simultaneousCapabilities field in CapabilityDescriptor. If the MC does not honor this capability by limiting the number of participants, the endpoint should ignore streams beyond its allowable range.

5. Proposed changes to H.245 (version dated July 6 1995)

5.1 Subset of H.245

5.1.1 Request PDUs not needed

The following subset of Request PDUs are not needed:

masterSlaveDetermination
openBiDirectionalChannelRequest
openBiDirectionalChannelRelease
multiplexEntrySend
multiplexEntrySendRelease
requestMultiplexEntry

5.1.2 Response PDUs not needed

The following subset of Response PDUs are not needed:

masterSlaveDeterminationAck
masterSlaveDeterminationReject
openBiDirectionalChannelAck
openBiDirectionalChannelReject
multiplexEntrySendAck
multiplexEntrySendReject
requestMultiplexEntryResponse

5.1.3 Indication PDUs not needed

The following subset of Indication PDUs is proposed:

jitterIndication

The jitterIndication PDU is not needed because RTCP provides this functionality.

The maximum allowable skew between two channels (ex. audio and video channels) can be used to determine the amount of buffering needed at the receiver. However, the source may not be able to compensate for the skew due to network delays on the two channels. In the case of multicast channels, it is much harder to satisfy all receivers of their skew requirements unless Quality-Of-Servide (QOS) can be provided by the network.

5.2 *Additional PDUs*

5.2.1 PDUs to distribute multicast information

Multicast information must be distributed to all endpoints involved in the conference. One new command PDU (ConferenceModelCommand), one new request PDU (requestConferenceModel), and one new response PDU (requestConferenceModelAck) are needed for this purpose. The ConferenceModelCommand PDU is sent by the MC to specify to the endpoints the model of the conference. If the conference model is multicast, multicast addresses and their associated port numbers and data types are also specified. This command may cause a switch between the centralized model and the multicast model. A switch will involve closing all existing logical channels and opening new ones. The requestConferenceModel PDU is sent by the endpoint to the MC to request the model of the conference. The requestConferenceModelAck PDU is sent by the MC to the endpoint, in response to the endpoint's request, to specify the model of the conference and its specifics.

```
ConferenceModelCommand                          ::=CHOICE
{
        centralized                               NULL,
        multicastTable                            SET SIZE (1..256) OF
MulticastTableEntry
}

requestConferenceModel                          ::=NULL
{
}

requestConferenceModelAck                       ::=CHOICE
{
        centralized                              NULL,
        multicastTable                           SET SIZE (1..256) OF MulticastTableEntry
}

MulticastTableEntry                             ::=SEQUENCE
{
        DataType                                 CHOICE
        {
               videoData                         NULL,
               videoControl                      NULL,
               audioData                         NULL,
               audioControl                      NULL,
               data                              NULL
        }
        multicastAddress                         OCTET STRING,
        portNumber                               INTEGER (0..65535)
       GuaranteedDelivery                       BOOLEAN
}
```

For example, if the MC has assigned two multicast addresses - one for video and another for audio - to a conference then it will send the following ConferenceModelCommand to all the endpoints involved:

```
ConferenceModelCommand
{
        multicastTable  = 4;
        MulticastTableEntry 1 = { DataType = audio, multicastAddress = 224.2.0.0,
                                   portNumber = 4000, GuaranteedDelivery = NO}
        MulticastTableEntry 2 = { DataType = audioControl, multicastAddress = 224.2.0.0,
                                   portNumber = 4001, GuaranteedDelivery = NO}
        MulticastTableEntry 3 = { DataType = video, multicastAddress = 224.2.4.0,
                                   portNumber = 4050, GuaranteedDelivery = NO}
        MulticastTableEntry 4 = { DataType = videoControl, multicastAddress = 224.2.4.0,
                                   portNumber = 4051, GuaranteedDelivery = NO}

}
```

In RTP/RTCP, even number port numbers are for RTP streams and the next odd number port for the corresponding RTCP streams. Therefore, in our example, port number 4000 will be used for the audio RTP packets, port number 4001 for audio RTCP packets, port 4050 for video RTP packets, and 4051 for video RTCP packets.

5.2.2 PDUs to select the MC when a switch is made from point-to-point to multipoint

A point-to-point conference can switch to multipoint if at least one endpoint is also capable of becoming an MC. If both endpoints can become an MC then the mcDetermination Request PDU is used to select the MC. The mcDetermination, mcDeterminationAck, and mcDeterminationReject PDUs works exactly the same way as the MasterSlaveDetermination, MasterSlaveDeterminationAck, and MasterSlaveDeterminationReject PDUs described in the H.245 recommendation.

```
mcDetermination                                    ::SEQUENCE
{
      statusDeterminationNumber                    INTEGER (0..4294967295)
}

mcDeterminationAck                                 ::SEQUENCE
{
      decision                                       CHOICE
       {
             master                                  NULL,
             slave                                   NULL
       }
}

mcDeterminationReject                              ::SEQUENCE
{
      cause                                          CHOICE
       {
             identicalNumbers                      NULL,
       }
}
```

5.3 *Extensions to the fields in the existing PDUs*

Changes to H.245 are required in order to extend it to include the multicast conference. These changes are shown in *italics* and **bold**.

5.3.1 Extend TerminalCapabilitySet Request PDU

The endpoint must be capable of participating in a centralized conference. Additionally, the endpoint could be capable of a multicast conference. Also, the endpoint could be capable of becoming a centralized conference MC and/or a multicast conference MC in addition to assuming the role of a participant. These capabilities are expressed as extensions to the TerminalCapabilitySet PDU. A multicast-capable endpoint does the assume an end-to-end multicast infrastructure. If the terminal does not receive multicast streams, it should present its new capabilities to the MC.

```
TerminalCapabilitySet                     ::=SEQUENCE
{
      sequenceNumber                      SequenceNumber,
       protocolIdentifier                  OBJECT IDENTIFIER,
       multiplexCapability                  MultiplexCapability OPTIONAL,
       capabilityTable                       SET SIZE (1..256) OF CapabilityTableEntry
OPTIONAL,
         multicastConference                BOOLEAN OPTIONAL,
         centralizedConferenceMC           BOOLEAN OPTIONAL,
         multicastConferenceMC             BOOLEAN OPTIONAL
}
```

5.3.2 Extend OpenLogicalChannel Request PDU

Two data types (video control and audio control) are added to include the video RTCP and the audio RTCP channels. The h22zLogicalChannelParameters is added to specify the fields required in the LAN protocols.

```
DataType                                    ::=CHOICE
{
        nonStandard                        NonStandardParameter,
        nullData                            NULL,
        videoData                           videoCapability,
        audioData                           audioCapability,
        data                                  DataApplicationCapability,
        encryptionData                     encryptionMode,
        videoControl                         NULL,
        audioControl                          NULL
}

LogicalChannelMultiplexParameters          ::=CHOICE
{
        h222LogicalChannelParameters       222LogicalChannelParameters,
        h223LogicalChannelParameters       223LogicalChannelParameters,
        h22zLogicalChannelParameters         h22zLogicalChannelParameters
}

h22zLogicalChannelParameters                 ::=SEQUENCE
{
        networkAddress                       OCTET STRING,
        portNumber                           INTEGER (0..65535),
        GuaranteedDelivery                  BOOLEAN
}
```

The use of logical channels requires an explanation. Each logical channel is identified by a unique logical channel number (logicalChannelNumber). The logical channel numbers are used by the conference control applications to refer to the different physical transport connections by names. These logical channel numbers are known only within the conference control applications of the endpoints involved in a conference; any application (audio, video, data) other than the control application does not understand logical channel numbers.

In the centralized conference, each logical channel maps to a unique physical transport connection between the endpoint/gateway and the MC. In the multicast conference, many logical channels between the endpoints/gateways and the MC will map to a single physical multicast transport connection. Even though the logical channels are opened between the MC and the endpoint, the physical data flow for a multicast transport connection is not from the endpoint to the MC but essentially a bus from which any endpoint can transmit or receive data. The MC opens logical channels to all capable endpoints, capability determined during capability exchange, in the conference to give them the ability to receive audio/video on the specified physical multicast transport connection. Those endpoints that wish to transmit, open a logical channel to the MC for the specified physical multicast transport connection.

Let us consider the following four examples pertaining to opening logical channels that map to the physical multicast transport connections. This example builds up from the previous example given for the ConferenceModelCommand. There are four physical multicast transport connections - two transport connections for video RTP and video RTCP, and two for audio RTP and audio RTCP - that map to four logical channels for transmitting and four for receiving between the MC and the endpoint:

1. If all the endpoints are capable of sending and receiving audio and video streams the MC will open four logical channels with each endpoint to allow the endpoints to receive audio and video. Each endpoint will open four logical channels with the MC to transmit audio and video.
2. If the endpoint is receive-only, it will not open any logical channel with the MC but the MC will open four logical channel to it.
3. If the endpoint is transmit only, it will open four logical channel to the MC but the MC will not open logical channels to it.
4. If the endpoint wishes to transfer an additional video on a separate multicast address to all the other endpoints then it will have to pick a unique multicast address and two port numbers and

open two logical channel (video RTP and video RTCP) with the MC. If the MC acknowledges the request, it will then send to all the endpoints the ConferenceModelCommand with the updated table of physical multicast transport connection information. The MC will then open the logical channels with all capable endpoints. The endpoints can, however, reject the request if they are not interested in the channels.

5.3.3 Extend OpenLogicalChannelReject Response PDU

There is a possibility that the receiver is already using the portNumber that the transmitter is specifying in the OpenLogicalChannel PDU. Therefore, in the centralized conference the receiver should be able to reject the OpenLogicalChannel PDU with the cause being that the port number is not available (portNumberNotAvailable) and it should specify the new port number that is acceptable. The transmitter can then send the OpenLogicalChannel PDU again with the port number supplied by the receiver. The transmitter should not expect the receiver to reserve the new port number but expect its OpenLogicalChannel PDU to be rejected again until finally it will get acknowledged.

In the multicast conference, the endpoint should use the multicast addresses and the port numbers supplied by the MC through the ConferenceModelCommand. But both the endpoint and the MC can reject the OpenLogicalChannel PDU if the multicast address or the port number is not appropriate. These four cases and their interpretations are as follows:

1. The MC rejects the OpenLogicalChannel PDU from the endpoint with the cause being multicastAddressInvalid if the MC does not allow the endpoint to pick a multicast address. The endpoint should use the multicast address supplied by the MC or it will not be able to transmit.
2. The MC rejects the OpenLogicalChannel PDU from the endpoint with the cause being portNumberNotAvailable if the MC does not allow the endpoint to pick a port number. The endpoint should use the port number supplied by the MC or it will not be able to transmit.
3. The endpoint rejects the OpenLogicalChannel PDU from the MC with the cause being multicastAddressInvalid if the MC has not picked a unique multicast address. If the MC does not fix this problem, this endpoint will not be able to receive.
4. The endpoint rejects the OpenLogicalChannel PDU from the MC with the cause being portNumberNotAvailable if it cannot close other applications that are using the same port number. If the MC does not fix this problem, the endpoint will not be able to receive.

```
OpenLogicalChannelReject                    ::=SEQUENCE
{
    logicalChannelNumber                    logicalChannelNumber
    cause
    {
            unspecified                     NULL,
            dataTypeNotSupported            NULL,
            dataTypeNotAvailable            NULL,
            unknownDataType                 NULL,
            multicastAddressInvalid         NULL,
            portNumberNotAvailable          INTEGER(0..65535)
    }
}
```