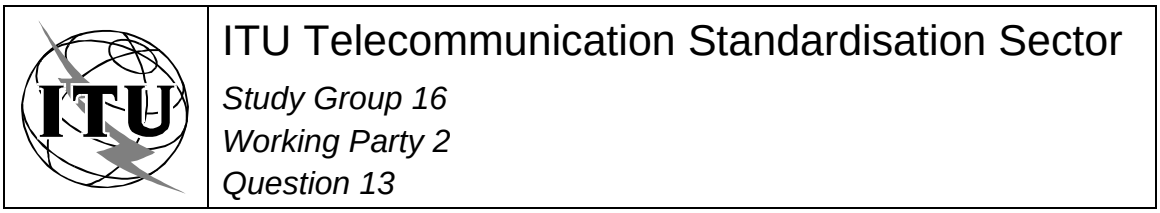


ITU Telecommunication Standardisation Sector

Study Group 16
Working Party 2
Question 13

Title: ***H.323 Annex E: "Call Signalling over UDP"***
Source: ***Editor***
Revision: ***4.1***
Status: ***Determined (Sept. 1998)***
Contact: ***David Gurle,*** ***david_gurle@vocaltec.com***

Table of Contents

E.1. Introduction

H.323 version 2 introduces the concept of ”Fast Connect”, which allows media cut-through in as little as in 2 round trips from callee to caller (including TCP messages), and in 2.5 round-trips from caller to callee.

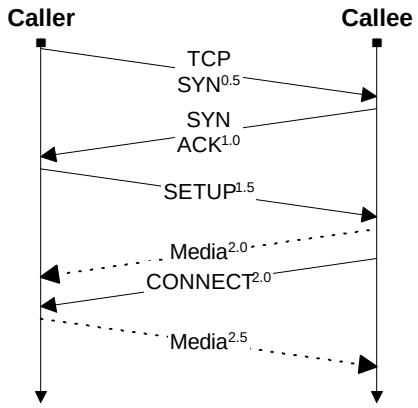
This can be reduced to 1^{rt} and 1.5^{rt} respectively by using UDP as the transport for H.323 messages, instead of TCP. After adding the required functionality to set-up a call over UDP, the procedure can be extended slightly to work for the full range of call-signalling messages. This is especially important when using the Gatekeeper-Routed-Model.

E.2. Call Setup Using UDP

E.2.1 H.323v2 Fast Connect Procedure

H.323 version 2 uses the TCP transport to carry H.225.0 messages, which means the least number of round trips possible to get media cut-through is 2 from Callee to Caller, and 2.5 from Caller to Callee.

Image/Table 1.: Information Flow for H.323 version 2 FastConnect.

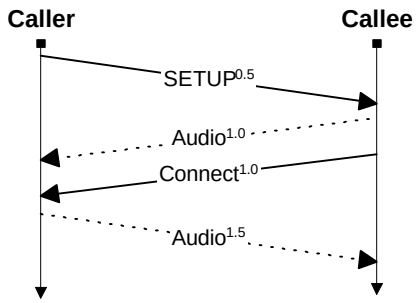


NOTE: Note that some messages in the TCP handshake procedure has been omitted for clarity.

E.2.2 UDP-Based Procedure

To get faster media cut-through, it is possible to use UDP for call-signalling transport, which effectively enables media cut-through in a single round trip:

Image/Table 2.: Information Flow for UDP-based Call-Setup

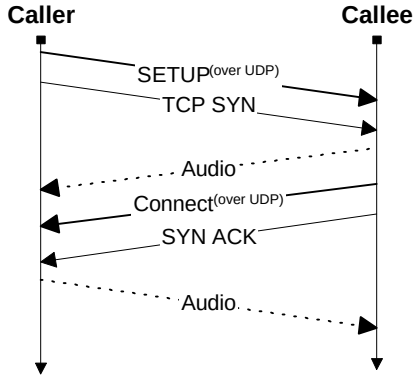


Applications should retransmit a lost packet if it does not get a reply after some time. The precise retransmission procedure is detailed later in this document.

E.2.3 Mixed TCP & UDP Procedure

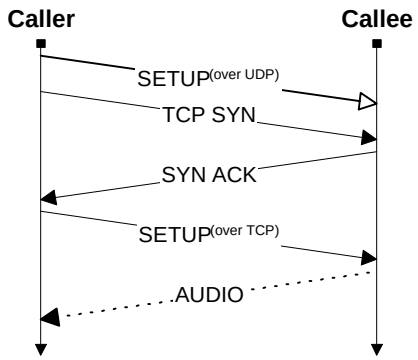
The procedures for TCP-based and UDP-based call setup are not mutually exclusive. Either UDP-based call setup may be attempted prior to TCP-based call setup or UDP-based and TCP-based call setup may be carried out in parallel. If the UDP-based setup succeeds, the TCP connection shall be released. If UDP-based call setup fails (because the remote peer does not support the Annex E procedures) the TCP-based setup shall be used — thus acting as fallback. As soon as the TCP-based call setup has completed, UDP-based communication shall no longer be used.

Image/Table 3.: Information Flow for Mixed TCP & UDP Procedure



This insures that if the UDP procedure fails, usual TCP-based procedures can take over immediately:

Image/Table 4.: Information Flow when UDP is not supported



This means that backwards compatibility when calling H.323 version 1 or 2 entities is transparent, because the v1/v2 application will not be aware of the UDP packet.

NOTE: It is recommended for applications that initiate a call and do not know if the remote side supports Annex E operations to use the procedure detailed above. If the calling application knows by some means that the remote callee supports UDP-based operations, it may use the procedure detailed in E.2.2.

E.3. Support for TCP Procedures

Annex E compliant devices shall support the full TCP-based procedures defined in H.323v2 to provide backward compatibility. Annex E compliant devices shall use the procedure described in section E.2.3, to insure seamless interoperability with H.323v1 and v2 entities.

To provide seamless compatibility with non-Annex E supporting devices, compliant devices should open a backward-compatible TCP H.225.0 call signalling connection in parallel to or shortly after attempting the UDP-based procedure. The UDP packet should be sent before attempting to open the TCP channel.

E.4. Annex E Transport

E.4.1 Supported H.323 Messages

Annex E provides a multiplexed transport layer that can be used to transmit various protocols. Often-used protocols such as H.225/Q.931, RAS, and H.245 have specific code points (also called “payload types”). Other protocols can be carried using the non-standard payload type, using an Object Identifier (OID) to identify the protocol.

NOTE: This version of Annex E specifies only the use of H.225.0 call-signalling messages (Q.931 like messages) and tunnelled H.245 messages (as defined in the H.323v2 suite). The use of RAS or other protocols using Annex E transport is left unspecified, and is for future study.

E.4.2 Ack Message

UDP messages can get lost. If the application needs assurance that a sent message arrived successfully, It may request an Ack message for the PDU.

A sender shall specify in the <ackRequested> field whether it wants to receive an Ack PDU for a PDU being sent, and the receiver shall reply with an Ack PDU if the <ackRequested> field is set.

Sending an message that requires an Ack means that no further messages that require an Ack shall be sent on the same call signalling channel until an Ack is received.

NOTE: Applications can create a completely reliable transport by setting the <ackRequested> field of every message and waiting until the Ack message is received. This however will be very much like TCP operations, with the exception of the application controlling the retrasmittion timers more correctly for real-time operations then TCP does.

NOTE: Annex E allows implementers to choose which messages are requiring an Ack. H.245 messages may not need an Ack, as they use a Request-Reply sequence (with some exceptions). This is also true for RAS messages (e.g. An RRQ is sent, and if an RCF or RRJ are not returned within the timeout, the messages is repeated).

This however means Annex E transport could be intertwined with the H.323 state machine, which may be undesirable. The decision if to use Annex E as a separate transport under H.323 or to depend on the H.323 state-machine is left to the discretion of the implementers.

E.4.3 Nack Message

A Nack message shall be used to signify some error condition. Such errors may be the inability to support a specific payload type, the arrival of a malformed PDU, and others. These messages may or may not have the effect of dropping an on-going call.

E.4.4 I-Am-Alive Optional Message

When running over TCP, the presence of a persistent TCP connection can insure that one side is aware of the remote side failures (by observing TCP failures). When running over UDP, there is no such “state” associated, and another procedure must be used.

The solution is for the master side of the call (as decided using the Master-Slave negotiation) to send an “I-Am-Alive” message to the slave side, to let the remote application know the host is still up. The slave will answer with an Ack messages as proof that it too is up.

The retransmission timer of the I-Am-Alive messages shall be reset on receiving any other call-signalling (i.e. not RTP) message over UDP, as it is proof the remote end is alive. This saves bandwidth, as I-Am-Alive messages will be sent only when required (i.e when no other call-signalling message is received).

Generating I-Am-Alive messages is optional, however, all entities shall support the ability to reply to I-Am- Alive messages (i.e. the ability to answer an I-Am-Alive message is not optional, and when such a message is received, it shall be answered according to the procedures defined in this annex).

NOTE: Some network operators may not wish to have endpoints generate I-Am-Alive messages, because they do consume some network bandwidth. I-Am-Alive messages are optional for this reason.

E.4.5 Sending Multiple Messages in a single PDU

It is possible to carry multiple messages per call and for multiple calls (i.e. multiple CRVs). It is possible for example to send multiple SETUP messages in a single UDP PDU. This is especially useful when two gateways are communicating (also referred to as trunking).

NOTE: There is no implicit relation between messages when they arrive in the same PDU.

E.5. Specifics

E.5.1 Choosing a Mode of Operation

During the initial exchange of messages between two peers, the transport mode of operation - TCP-based or UDP-based - is chosen according to the procedures specified in the following.

After a mode of operation is chosen (e.g. UDP or TCP), only that mode shall be used for the remainder of the call. If both peers choose to use UDP then the TCP connection shall be dropped; this shall not be interpreted as termination of the call. If both peers choose to use TCP, transmission of any PDUs via UDP shall be discontinued and all associated timeouts shall be cancelled.

E.5.2 Initiating a Call: Calling Side

The endpoint wishing to call another endpoint shall first complete the gatekeeper procedures as specified in H.323v2. Then the calling endpoint may decide whether or not to proceed with conventional (H.323v1 and H.323v2) call setup, or with UDP-based call setup. For UDP-based call setup the following procedures shall apply.

The caller shall create a H.225.0v2 call setup PDU, encapsulate it and transmit it in a UDP packet to the callee (or the Gatekeeper if using the Gatekeeper routed model). It shall then start two timers, **T1** and **T4**.

If **T1** expires (because an Ack message was not received) before a response packet matching the SETUP was received from the callee (or the Gatekeeper, and which may be any of ACK, CONNECT, ALERTING, FACILITY, or CALL PROCEEDING), the sender shall retransmit the SETUP packet (following the procedures defined below) and start a timer **T3**. If **T3** expires, another retransmission shall be performed, and **T3** shall be restarted. After a total of **N1** transmissions, the caller shall stop retransmitting and revert to the exclusive use of TCP call signalling connection instead of UDP.

In parallel, the caller shall wait until **T4** expires, then initiate setup of a TCP call signalling connection to the same address to which the UDP packet was sent to before.

The transport (TCP or UDP) via which the first response for a particular call (identified by means of the conference identifier field) - e.g. CALL PROCEEDING - is received determines the transport mode for the call.

If UDP has been chosen as transport mode and the caller has opened a TCP connection to the peer then the caller shall close the TCP connection.

If no response is received via UDP and the TCP connection establishment failed, the call has failed (destination unreachable).

E.5.3 Initiating a Call: Called Side

The called endpoint shall perform all necessary gatekeeper procedures as defined in H.323v2.

The calling entity shall be prepared to accept incoming calls via UDP packets as well as via TCP-based call signalling connections. It shall always (attempt to) reply to an H.225.0 call signalling message using the transport (TCP or UDP) it was received.

The transport (TCP or UDP) via which the SETUP for a particular call (identified by means of the conference identifier field) message is received determines the transport mode for the call.

If a SETUP message is received via TCP and an identical SETUP has not been received via UDP then the corresponding call setup procedures defined in H.323 shall be followed.

If a SETUP message is received via UDP, the callee shall verify that the same request has not been received before via TCP or UDP — duplicates may be detected by means of the conference identifier field contained in every H.225.0 call signaling message. If the request is a

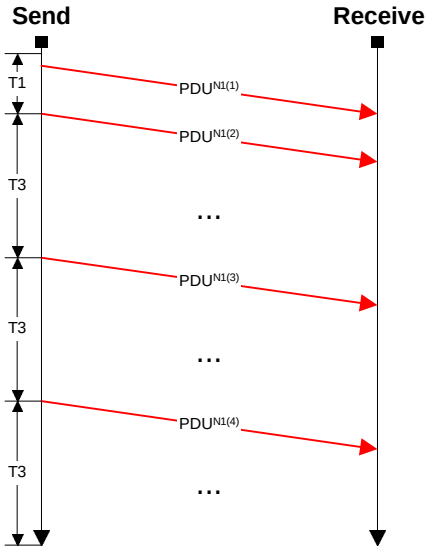
duplicate of a message formerly received via TCP, it shall be silently ignored. If the request is a duplicate of a message formerly received via UDP, the response given to the original request message shall be retransmitted immediately and the currently active timer (see below) shall be restarted. If the message is not a duplicate, the callee shall determine the appropriate actions and send the corresponding response via UDP (any of ACK, CALL PROCEEDING, ALERTING, CONNECT, or FACILITY). With the first transmission of the response, the callee shall start a timer **T1**.

If **T1** expires (because an Ack message was not received), the callee shall retransmit the packet and start a timer **T3**. If **T3** expires, the callee shall send another retransmission of the response message and restart **T3**. After the response message has been sent for a total of **N1** times, the callee shall stop retransmitting the message and start a timer **T5**. After **T5** expires, the callee shall discard the conference / call identification information and associated state, and regard the setup of this session as failed.

E.5.4 Sender Retransmission Policy

A sender shall use the following retransmission policy. If it does not get a reply within **T1**, it should retransmit the PDU to the sender, and repeat sending up to **N1** total PDUs with an interval of **T3**.

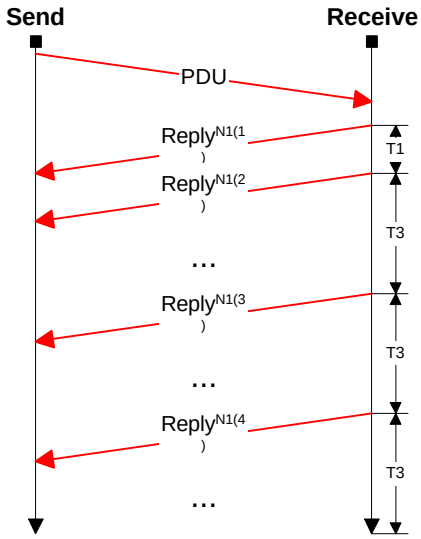
Image 5.: Sender Retransmission Information Flow



E.5.5 Receiver Retransmission Policy

When the receiver receives a PDU over UDP, it shall return a reply PDU over UDP. If it does not get a reply within **T1**, it should retransmit the PDU to the sender, and repeat sending up to **N1** total PDUs with an interval of **T3**.

Image 6.: Receiver Retransmission Information Flow



E.5.6 UDP Well-Known-Port

The well-known port registered with IANA to allow H.323 entities to listen to incoming TCP call signalling connections (1720) has been already registered with IANA for H.323 use on the UDP port-space, so can be re-used for Annex E operations. Devices may transmit from any random port.

E.5.7 Sender Sequence Number Policy

Assigned per host-address + source-port, Sending applications shall start with some random value, incrementing by 1 for every PDU sent. If the sequence number reaches 2^{24} (16,777,216) it shall wrap around to 0.

E.5.8 Receiver Sequence Number Policy

When receiving a UDP packet, the application should check the host-address + source-port + sequence number to recognise duplicate messages. The application may recognise packet-loss when finding gaps in sequence numbers.

E.5.9 Timers and Counters

The following timers and counters should be used by both the sender and the receiver of messages:

Image/Table 7.: Timers & Counters

Item	*Value*	*Comments*
T1	1000 ms	A reasonably small value is chosen here to compensate for possible 1st packet loss, and still not delay call setup by too much.
T2	*reserved*	-
T3	3000 ms	If the first packet is lost, apply some back-off.
T4	< (T1+(T3*(N1-1)))	Time between the initial UDP packet is sent and when attempting to open the TCP H.323 connection.
T5	<= 30 seconds	Time to wait after the last transmission of an call-signalling or I-Am-Alive message via UDP before deleting the associated call state (and declaring the call failed). Reception of I-Am-Alive or any other call-signalling message shall reset this timer.
T6	*reserved*	-
T7	6 seconds	I-Am-Alive transmission interval.
T8	*reserved*	-
N1	4	If no packet came back within 10 seconds after 4 retransmissions, it is probably a sub-optimal network and it is futile to continue.

E.5.10 Timers & Counters Implementation Note

The above timer values may be adjusted to match certain call scenarios if the expected round-trip and processing delays are well-known and the messages are not sent across the common wide area Internet. This applies, for example, for use in intranets as well as for trunking in a closed environment, where all the network resources are (administratively) controlled by the enterprise.

Implementations shall be sensible about the effects too short timeouts may have on the network infrastructure and shall be able to be explicitly configurable to match the timer values defined above.

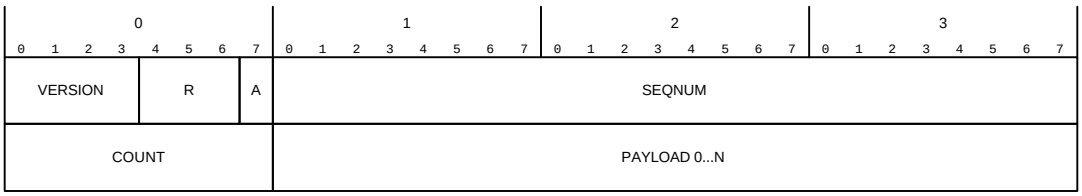
E.6. Wire Protocol

Annex E transport uses binary encoding as defined in the rest of this section. Structures are network-aligned (e.g. big-endian).

E.6.1 Header Structure

The following structure shall be used to encode the Header part of the Payload.

Image/Table 8.: Header Structure



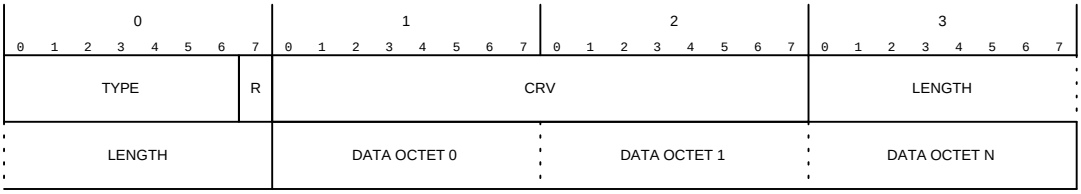
Image/Table 9.: Content of Fields

Field	*Contents*	*bits*
VERSION	Unsigned Integer between 0..15: senders shall set this field to zero. Version number 15 is reserved for experimental use and shall be ignored by commercial implementations.	4
R	3 Bits: Reserved for future use, shall be set to zero by sender, and shall be ignored by the receiver.	3
A	Boolean: TRUE indicates that an Ack requested for this PDU; shall not be set to TRUE for PDUs that contain only an ACK or a NACK structure	1
SEQNUM	Unsigned Integer between 0 and 16,777,215: the sequence number of this PDU	24
COUNT	Unsigned Integer between 0..255: the number of payload structures in this PDU. 0 means there is 1 payload, 1 means there are two, and so on.	8
PAYLOAD	Sequence of payload structures	n

E.6.2 Payload Structure

The following structure shall be used to encode Annex E payloads.

Image/Table 10.: Payload Structure



Image/Table 11.: Content of Fields

Field	Contents	*bits*
TYPE	Unsigned Integer: the type of the payload, as defined in Table 12.	7
R	Reserved for future use. Shall be set to zero by senders, and shall be ignored by receivers.	1
CRV	Unsigned Integer: The Call this payload belongs to. The CRV value shall conform to the structure as specified in Q.931. Specifically, the call reference flag shall be included as the most significant bit of the CallReferenceValue. This restricts the actual CRV to the range of 0 through 32767, inclusive.	16
LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data.	16
DATA	The actual payload data.	n

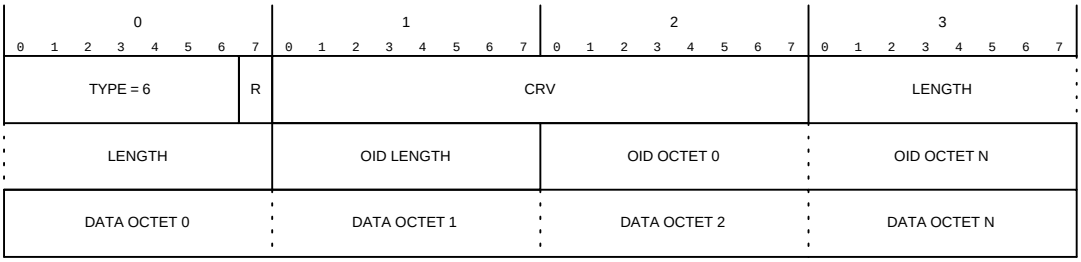
Image/Table 12.: Content of <TYPE> Field

Value	*Meaning*
0	reserved *(may be used in the future for H.225.0 RasMessage PDUs)*
1	octet-stream contains an H323-UU-PDU from H.225.0.
2	reserved *(may be used in the future for H.245 PDUs)*
3	octet-stream contains a I-Am-Alive structure
4	octet-stream contains a Ack Structure
5	octet-stream contains a Nack Structure
6	octet-stream contains a Non-Standard payload structure as defined in section E.7.3
7..126	Reserved for future use

E.6.3 Non-Standard Payload structure

The following structure shall be used to encode Annex E non-standard payloads. The data part of the payload shall begin with the following structure, and continue with the actual payload data, as specified in the <Payload.Length> field.

Image/Table 13.: Non-Standard Payload Structure



Image/Table 14.: Content of Non-Standard Structure Fields

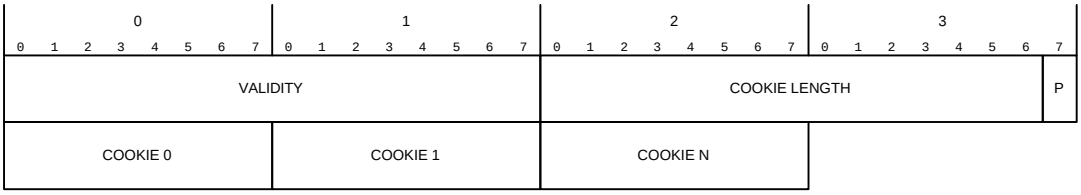
Field	Contents	*bits*
OID LENGTH	Unsigned Integer: The length (in OCTETs or BYTEs) of the OID	16
OID OCTET	The actual OID	n

E.6.4 I-Am-Alive Structure

The following structure shall be used to encode Annex E I-Am-Alive payloads. The validly period is expressed in 100s of milliseconds.

- If the replyRequested field is set to TRUE, the receiver shall reply with an I-Am-Alive of it's own with the cookie (if provided). When replying, the replyRequested field shall be set to FALSE.
- replyRequested is not the same as ackRequested in the PDU header, which results in an Ack Message. replyRequested results in an I-Am-Alive Message.
- If a validity period is set to ZERO (0), timer T7 shall be used.

Image/Table 15.: I-Am-Alive Structure



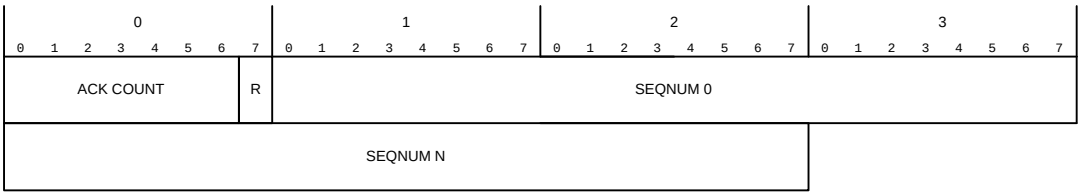
Image/Table 16.: Content of Non-Standard Structure Fields

Field	*Contents*	*bits*
VALIDITY	Unsigned Integer: The time in 100s of Milliseconds that this I-Am-Alive is valid for.	16
COOKIE LENGTH	The length (in BYTEs ot OCTETs) of the COOKIE field	15
P	Reply Requested	1
COOKIE	BYTEs or OCTETs of the cookie	n

E.6.5 Ack Structure

The following structure shall be used to encode Annex E Ack payloads:

Image/Table 17.: Ack Structure



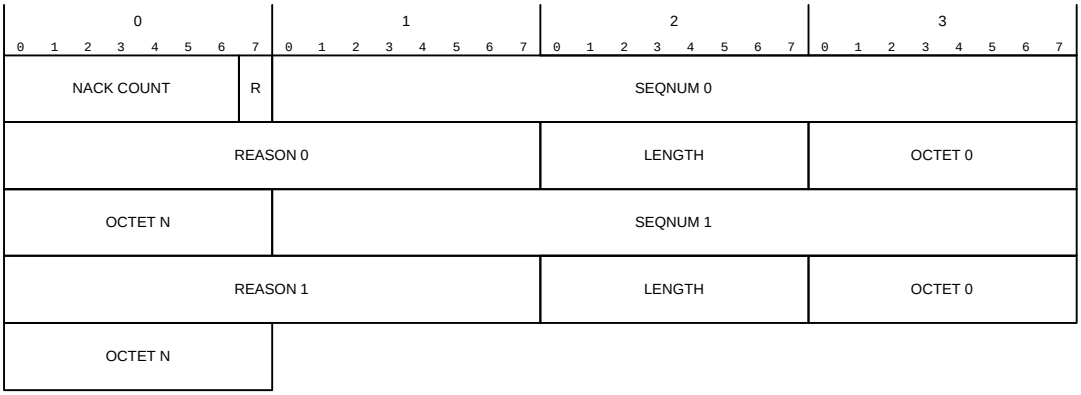
Image/Table 18.: Content of Fields

Field	*Contents*	*bits*
ACK COUNT	The number of SEQNUM fields that follow	7
R	Reseverd	1
SEQNUM 0..N	The Sequence Numbers of the PDUs that are being ACKed for	24 x n

E.6.6 Nack Structure

The following structure shall be used to encode the Nack structure:

Image/Table 19.: Nack Structure



Image/Table 20.: Content of Fields

Field	*Contents*	*bits*
NACK COUNT	The number of SEQNUM fields that follow	7
R	Reseverd for future use	1
SEQNUM 0..N	The Sequence Numbers of the PDUs that is being NACKed for	24 x n
REASON 0..N	The reason (matching the Nack sequence number) for the NACK	16 x n
LENGTH 0..N	Length of Nack-specific data	8 x n
OCTETs	Nack-specific data octets	n

Image/Table 21.: Nack Reasons

Value	*Meaning*	*Length*	*Data*
0	payload type not supported	8 bits	Unsigned Integer; Payload as defined in Table 18
1	Non-Standard payload type not supported	n	OID
1.. 32767	Reserved for future use		

E.7. Open Issues (Informative)

E.7.1 Motivation Section

Problem:	Should the "Motivation" section be left in the Annex?
Solution:	Mark the section as informative. It is felt that a discussion of why UDP operations are required should be left in the document.
Status:	Closed

E.7.2 I-Am-Alive Interval Calculations

Problem:	Are I-Am-Alive messages required?
Solution:	I-Am-Alive messages are now optional.
Status:	Closed

E.7.3 I-Am-Alive Initiator

Problem:	Do both sides of a call send I-Am-Alive messages or only the master of a call, making the slave answer (by setting the replyRequested field to TRUE)
Solution:	Make the master of a call the I-Am-Alive message initiator.
Status:	Closed

E.7.4 Use of ASN.1 for Annex E Structures

Problem:	It is possible to use a simpler encoding then ASN.1 PER for the Annex E structures, one that does not require an ASN.1 PER codec to parse. This may improve performance and simplify implementations. This is possible because all H.323 messages are encoded into an "octet stream", and Annex E does not require any part of H.323 ASN to compile.
Solution:	Use a fixed format header instead of ASN.1 PER.
Status:	Closed

E.7.5 Can RTP PDUs be considered as I-Am-Alive Refreshers?

Problem:	The I-Am-Alive message is used to reset the I-Am-Alive timer. Do RTP packets arriving from the same host (same IP address and call state) be used?
Solution:	No. Only call-signalling messages are considered I-Am-Alive timer refresher.
Status:	Closed

E.7.6 Is a Generic Nack message required?

Problem:	Annex E adds the Ack message. Do implementers need also a Nack message to signify some errors? (such as "payload-type-not-supported, but call was not dropped")
Solution:	Yes. Nack may be used to signal error-states that do not require a dropped-call.
Status:	Closed

E.7.7 Is a Sequence-Number range of 16-bits required?

Problem:	Two bytes are used in every UDP header for the sequence number field. Is this really required? Two bytes were selected for parity with RTP.
Solution:	Sequence-Number extended to contain 3 BYTEs/OCTETs.
Status:	Closed

E.7.8 Is a maximum I-Am-Alive validity period of 65 seconds enough?

Problem:	Using 16-bit value for millisecond, the maximum validity period for I-Am-Alive messages is 65 seconds. This may not be enough for very large conferences.
Solution:	the validity is expressed using 100s of milliseconds. This allow the `<validityPeriod>` field to communicate periods up to 1.8 hours.
Status:	Closed

E.7.9 Should Sequence-Numbers begin at 0?

Problem:	Do we enforce devices to always start sequence numbers at zero? Starting at Zero makes debugging a little simpler, as "new" sessions can be recognised quickly.
Possible Solutions:	Sequence numbers shall start at some random number.
Status:	Closed

E.8. Motivation (Informative)

The motivation for this approach was described during the SG16 Geneva meeting in January 1998.

E.8.1 Number of round-trip times before media transmission.

This refers to the known problem of H.323v1 and is solved in part by using the fast connect procedures of H.323v2. Using H.323 over UDP can further improve the solution.

E.8.2 TCP vs. UDP

What: Because it run over TCP, H.323 call-signaling performance suffers greatly when running on congested networks, or on networks that maintain a higher priority for UDP traffic then they do for TCP traffic.

Why: Two items have to be named here:

- TCP requires an extra round trip for connection establishment before any data can be sent.
- TCP obeys its own retransmission strategies which cannot easily (and should not) be altered by an application. For example, the default timeouts of TCP incur a delay of several seconds if the initial TCP packet is lost. In addition, further algorithms (such as TCP slow start) make TCP adapt defensively to observed network congestion (measured e.g. by timeouts). Using UDP as a transport gives the application full control over the retransmission scheme employed; in particular, the timeouts can be adjusted to deal with loss of the first packet.

Ideas: A possible solution to this problems seems to be an alternative method of sending a UDP packet to the target carrying call-signaling information.

E.8.3 Packet-size

What: A too large packet size of the H.225 SETUP message may further delay the connection setup by one or more round-trip - depending on how many TCP segments are needed to carry the data.

Why: The way TCP works: the slow start algorithm (RFC 2001) which is used in most TCP implementations allows a TCP connection only to send a single segment of bytes (the size of a segment is negotiated during connection setup and is typically less than the expected MTU for a network; the minimum MTU for IPv4 is 576 bytes).

After this, the sender has to wait for the acknowledgement before it may transmit segments #2 and #3. Therefore, if a Setup message does not fit into the first segment, at least an additional round-trip time will pass before the recipient has got all the data and can start processing.

Idea: Ensure that the setup packet does not grow larger than IP MTU, i.e. do not stuff every conceivable data structure into the SETUP and CONNECT messages.

E.8.4 Fail-Over Architecture

What: When a persistent H.225 link fails, it is difficult to switch it over to another host without dropping calls. For example, it is difficult to create a backup gateway that will be used if the main gateway fails without disturbing on-going calls.

Why: TCP does not have a fail-over mechanism.

Idea: carry signaling messages over UDP, as the sender can just send to another (backup) machine when the primary host fails (to respond). In effect, we make H.323 more reliable by using UDP.

E.8.5 Scalable Architecture

What: The need to maintain 2 separate TCP session for every H.323 call (i.e. one for H.225 and one for H.245) is improved somewhat by using H.245 tunneling, introduced in H.323v2. When attempting to use the Gatekeeper-routed-Model however, we are forced to use a large number of TCP "legs" to form a call.

Why: TCP requires a stateful "session" to work.

Idea: carry signaling messages over UDP. This allows building more scalable Gateways and Gatekeepers

…END…