Network Working Group
Internet-Draft
Updates: 4582, 4583
(if approved)
Intended status: Standards Track
Expires: September 3, 2010

M. Thompson, Ed.
T. Kristensen
G. Sandbakken
T. Andersen
E. McLeod
TANDBERG

March 2, 2010

Revision of the Binary Floor Control Protocol (BFCP) for use over an unreliable transport
draft-sandbakken-xcon-bfcp-udp-02

Abstract

This memo extends the Binary Floor Control Protocol (BFCP) for use over an unreliable transport. It details a set of revisions to the protocol definition document and the specification of BFCP streams in the Session Description Protocol (SDP).

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 3, 2010.

Copyright Notice

Table of Contents

1. Introduction

The motivation for using unreliable transports for BFCP [RFC4582] messages is fuelled by network deployments where RTP proxies are present for NAT and firewall traversal. In these deployments, TCP may neither be applicable nor appropriate, for example, due to lack of support for TCP media relay or ICE-TCP [I-D.ietf-mmusic-ice-tcp].

This memo extends the BFCP protocol to support unreliable transport. Minor changes to the transaction model are introduced in that all requests now have an appropriate response to complete the transaction. The requests are sent with a retransmit timer associated with the response to achieve reliability.

The intension is not to change the semantics of BFCP, but to present a trivial and workable extension that permits UDP as a transport. Existing implementations in the spirit of the approach detailed in -00 and -01 of this draft have demonstrated the approach to be feasible. The purpose of this document is to formalise the deviations from the baseline specification enabling interoperability between implementations.

The content of this draft relates to the BFCP protocol specification [RFC4582] and the format for the specification of BFCP streams in the SDP [RFC4583]. This memo is written with the goal of being incorporated into an upcoming revision of those documents without requiring additional protocol and stream specification documents.

This draft is not recommended for adoption as an XCON working group item at this time owing to the outstanding work detailed in Section 7, but is submitted for information and discussion within the XCON community.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Revision of RFC4582

This section details revisions to [RFC4582], the base protocol specification of BFCP. The section number to which updates apply are indicated in parentheses in the titles of the sub-sections below.

3.1. Overview of Operation (4)

Fourth paragraph change:

> There are two types of transaction in BFCP: client-initiated transactions and server-initiated transactions. Client-initiated transactions consist of a message from a client to the floor control server and a response from the floor control server to the client. Correspondingly, server-initiated transactions consist of a message from the floor control server to a client and the associated acknowledgement message from the client to the floor control server. Both messages can be related because they carry the same Transaction ID value in their common headers.

3.2. Floor Participant to Floor Control Server Interface (4.1)

Before seventh paragraph (page 9), insert:

> Figures 2 and 3 below show call flows for two sample BFCP interactions when used over reliable transport. [Appendix A] (Note: here-in Section 5.16) shows the same sample interactions but over an unreliable transport.

3.3. COMMON-HEADER Format (5.1)

The figure below should replace Figure 5: COMMON-HEADER format.

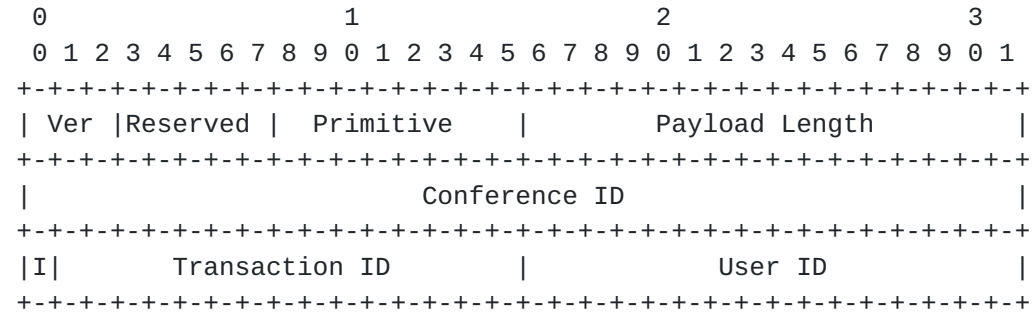


Figure 1: COMMON-HEADER format

The following text preceeds "Transaction ID" on page 16:

> I: The Transaction Initiator (I) flag-bit has relevance only for use of BFCP over unreliable media. When clear, it signifies that the transaction was opened by the client (floor participant, chair) and that the Transaction ID that follows has been generated by the client; when set, the transaction is a server-initiated transaction and the Transaction ID that follows is pertinent to

the floor control server. Where BFCP is used over reliable transports, the flag has no significance and SHOULD be cleared.

Note: An alternative is that we don't actually specify bit-16 of the Transaction ID to be a flag per se, but define a range of Transaction IDs that are defined as valid for each supported transport.

The description of Transaction ID should have the final clause deleted with the reference to Section 8 remaining. The value used for server-initiated transactions shall be non-zero when BFCP is used over unreliable transports, and this qualification shall be described in the updated Section 8.

The values below should be appended to the end of Table 1: BFCP primitives.

Value	Primitive	Direction
14	FloorRequestStatusAck	P -> S ; Ch -> S
15	ErrorAck	P -> S ; Ch -> S
16	FloorStatusAck	P -> S ; Ch -> S
17	Goodbye	P -> S ; Ch -> S ; S -> P ; S -> Ch
18	GoodbyeAck	P -> S ; Ch -> S ; S -> P ; S -> Ch

Table 1: BFCP primitives

3.4. ERROR-CODE (5.2.6)

The value below should be appended to the end of Table 5: Error Code meaning.

Value	Meaning
10	Unable to parse message

Table 2: Error Code meaning

3.5. FloorRequestStatusAck (5.3.14)

This new subsection should be added to specify the normative ABNF for the new primitive, FloorRequestStatusAck.

Floor participants and chairs acknowledge the receipt of a FloorRequestStatus message from the floor control server when communicating over unreliable transport. The following is the format of the FloorRequestStatusAck message:

```
FloorRequestStatusAck          =    (COMMON-HEADER)
                                   *[EXTENSION-ATTRIBUTE]
```

Figure 2: FloorRequestStatusAck format

3.6. ErrorAck (5.3.15)

This new subsection should be added to specify the normative ABNF for the new primitive, ErrorAck.

Floor participants and chairs acknowledge the receipt of an Error message from the floor control server when communicating over unreliable transport. The following is the format of the ErrorAck message:

```
ErrorAck                       =    (COMMON-HEADER)
                                   *[EXTENSION-ATTRIBUTE]
```

Figure 3: ErrorAck format

3.7. FloorStatusAck (5.3.16)

This new subsection should be added to specify the normative ABNF for the new primitive, FloorStatusAck.

Floor participants and chairs acknowledge the receipt of a FloorStatus message from the floor control server when communicating over unreliable transport. The following is the format of the FloorStatusAck message:

```
FloorStatusAck                 =    (COMMON-HEADER)
                                   *[EXTENSION-ATTRIBUTE]
```

Figure 4: FloorStatusAck format

3.8. Goodbye (5.3.17)

This new subsection should be added to specify the normative ABNF for the new primitive, Goodbye.

BFCP entities that wish to dissociate themselves from their remote participant do so through the transmission of a Goodbye. The following is the format of the Goodbye message:

```
Goodbye                    =    (COMMON-HEADER)
                               *[EXTENSION-ATTRIBUTE]
```

Figure 5: Goodbye format

3.9. GoodbyeAck (5.3.18)

This new subsection should be added to specify the normative ABNF for the new primitive, GoodbyeAck.

BFCP entities communicating over an unreliable transport should acknowledge the receipt of a Goodbye message from a peer. The following is the format of the GoodbyeAck message:

```
GoodbyeAck                 =    (COMMON-HEADER)
                               *[EXTENSION-ATTRIBUTE]
```

Figure 6: GoodbyeAck format

3.10. Transport (6)

Much of the existing text remains but demoted to become subsection 6.1. This draft recommends an additional behaviour for entities participating in communication over a reliable transport that either wish to leave or are asked to leave an established BFCP connection, as detailed in the revised section introduction text below.

Note: The UDP fragmentation handling issue is still unfinished as it is felt that the document should allow a mechanism for messages that can grow significantly (e.g. UserStatus) to be split into separate additive messages.

The transport over which BFCP entities exchange messages depends on how clients obtain information to contact the floor control server (e.g., using an SDP pffer/answer exchange [RFC4583]). Two

transports are supported: TCP, appropriate where entities can be sure that their connectivity is not impeded by NAT devices, media relays or firewalls; and UDP for those deployments where TCP may not be applicable or appropriate.

If a client wishes to end its BFCP association with a floor control server, it is RECOMMENDED that the client send a Goodbye message to dissociate itself from any allocated resources. If a floor control server wishes to end its BFCP association with a client (e.g., the Focus of the conference informs the floor control server that the client has been kicked out from the conference), it is RECOMMENDED that the floor control server send a Goodbye message towards the client.

3.11. Reliable transport (6.1)

BFCP entities may elect to exchange BFCP messages using TCP connections. TCP provides an in-order reliable delivery of a stream of bytes. Consequently, message framing is implemented in the application layer. BFCP implements application-layer framing using TLV-encoded attributes.

A client MUST NOT use more than one TCP connection to communicate with a given floor control server within a conference. Nevertheless, if the same physical box handles different clients (e.g., a floor chair and a floor participant), which are identified by different User IDs, a separate connection per client is allowed.

If a BFCP entity (a client or a floor control server) receives data that cannot be parsed, the entity MUST close the TCP connection, and the connection SHOULD be reestablished. Similarly, if a TCP connection cannot deliver a BFCP message and times out, the TCP connection SHOULD be reestablished.

The way connection reestablishment is handled depends on how the client obtains information to contact the floor control server. Once the TCP connection is reestablished, the client MAY resend those messages for which it did not get a response from the floor control server.

If a floor control server detects that the TCP connection towards one of the floor participants is lost, it is up to the local policy of the floor control server what to do with the pending floor requests of the floor participant. In any case, it is RECOMMENDED that the floor control server keep the floor requests (i.e., that it does not cancel them) while the TCP connection is reestablished.

To maintain backwards compatability with older implementations of

[RFC4583], BFCP entities MUST interpret the graceful close of their TCP connection from their associated participant as an implicit Goodbye message.

3.12. Unreliable transport (6.2)

BFCP entities may elect to exchange BFCP messges using UDP datagrams. UDP is an unreliable transport where neither delivery nor delivery order is assured. At most one BFCP message shall be conveyed per datagram. The message format for exchange of BFCP in UDP datagrams is the same as for a TCP stream above.

Clients MUST announce their presence to the floor control server by tranmission of a Hello message. This Hello message MUST be responded to with a HelloAck message and only upon receipt can the client consider the floor control service as present and available.

As described in [Section 8], each request sent by a floor participant or chair shall form a client transaction that expects an acknowledgement message back from the floor control server within a retransmission window. Concordantly, messages sent by the floor control server that are not transaction-completing (e.g. FloorStatus announcements as part of a FloorQuery subscription) are server-initiated transactions that require acknowledgement messages from the floor participant and chair entities to which they were sent.

If a BFCP entity receives data that cannot be parsed, the receiving participant MAY send an Error message with parameter value 10 indicating receipt of a malformed message. If the message can be parsed to the extent that it is able to discern that it was a response to an outstanding request transaction, the client MAY discard the message and await retransmission. BFCP entities receiving an Error message with value 10 SHOULD acknowledge the error and act accordingly.

Transaction ID values are non-sequential and entities are at liberty to select values at random. Entities MUST only have at most one outstanding request transaction at any one time. Implicit subscriptions, such as FloorRequest messages that have multiple responses as the floor control server processes intermediate states until Granted or Denied terminal states attained, can be characterised by a client-initiated request transaction whose acknowledgement is implied by the first FloorRequestStatus response from the floor control server. The subsequent changes in state for the request are new transactions whose Transaction ID is determined by the floor control server and whose receipt by the client participant shall be acknowledged with a FloorRequestStatusAck message.

By restricting participants to having at most one pending transaction open, the out-of-order receipt of messages is mitigated. A server-initiated request (e.g., a FloorStatusRequest with an update from the floor control server) received by a participant before the initial FloorRequestStatus message that closes the client-initiated transaction that was instigated by the FloorRequest clearly supercedes the information conveyed in the delinquent response. As the floor control server cannot send a second update to the implicit floor status subscription until the first is acknowledged, ordinality is maintained.

BFCP entities SHOULD ensure that their messages are smaller than the recommended MTU size of 1300 bytes when encoded to minimise likelihood of fragmentation en route to their peer entity.

If a BFCP entity receives an ICMP port unreachable message mid-conversation, the entity SHOULD treat the conversation as closed (e.g., an implicit Goodbye message from the peer) and behave accordingly. The entity MAY attempt to re-establish the conversation afresh. The new connection will appear as a wholly new floor participant, chair or floor control server with all state previously held about that participant lost.

Note: This is because the peer entities cannot rely on IP and port tuple to uniquely identify the participant, nor would extending Hello to include an attribute that advertised what the entity previously was assigned as a User ID be acceptable due to session hijacking.

In deployments where NAT appliances, firewalls or other such devices are present and affecting port reachability for each entity, peer connectivity checks, relay use and NAT pinhole maintenance SHALL be achieved through the mechanisms defined in [I-D.ietf-mmusic-ice].

4. Lower-Layer Security (7)

For review in future revisions of this draft, per Section 7.

5. Protocol Transactions (8)

The final clause of the introduction to section 8 shall be changed to read:

> Since they do not trigger any response, their Transaction ID is set to 0 when used over reliable transports, but must be non-zero and unique in the context of outstanding transactions over unreliable transports.

When using BFCP over unreliable transports, all requests will use retransmit timer T1 (see Section 5.2) until the transaction is completed.

5.1. Server Behavior (8.2)

The final clause of this section shall be changed to read:

Server-initiated transactions MUST contain a Transaction ID equal to 0 when BFCP is used over reliable transports. Over unreliable transport, the Transaction ID shall have the same properties as for client-initiated transactions: the server MUST set the Transaction ID value in the common header to a number that is different from 0 and that MUST NOT be reused in another message from the server until the appropriate response from the client is received for the transaction. The server uses the Transaction ID value to match this message with the response from the floor participant or floor chair.

5.2. Timers (8.3)

New section:

When BFCP entities are communicating over an unreliable transport, two retransmission timers are employed to help mitigate against loss of datagrams. Retransmission and response caching are not required when BFCP entities communicate over reliable transports.

5.2.1. Request retransmission timer, T1 (8.3.1)

T1 is a timer that schedules retransmission of a request until an appropriate response is received or until the maximum number of retransmissions have occurred. The timer doubles on each re-transmit, failing after three unacknowledged transmission attempts.

If a valid respone is not received for a client- or server-initiated transaction, the implementation MUST consider the BFCP association as failed. Implementations SHOULD follow the reestablishment procedure described in section 6 (e.g. initiate a new offer/answer [RFC3264] exchange). Alternatively, they MAY continue without BFCP and therefore not be participant in any floor control actions.

5.2.2. Response retransmission timer, T2 (8.3.2)

T2 is a timer that, when fires, signals that the BFCP entity can release knowledge of the transaction against which it is running. It is started upon the first transmission of the response to a request and is the only mechanism by which that response is released by the

BFCP entity. Any subsequent retransmissions of the same request can be responded to by replaying the cached response, whilst that value is retained until the timer has fired.

T2 shall be set such that it encompasses all legal retransmissions per T1 plus a factor to accommodate network latency between BFCP entities.

5.2.3. Timer values (8.3.3)

The table below defines the different timers required when BFCP entities communicate over an unreliable transport.

Timer	Description	Value/s
T1	Initial request retransmission timer	0.5s
T2	Response retransmission timer	10s

Table 3: Timers

5.3. Receiving a response [to a FloorRequest Message] (10.1.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a FloorRequest from a participant, the floor control server MUST respond with a FloorRequestStatus message within the transaction failure window to complete the transaction.

5.4. Receiving a response [to a FloorRelease Message] (10.2.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a FloorRelease from a participant, the floor control server MUST respond with a FloorRequestStatus message within the transaction failure window to complete the transaction.

5.5. Receiving a response [to a ChairAction Message] (11.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a ChairAction from a participant, the floor control server MUST respond with a ChairActionAck message within the transaction failure window to complete the transaction.

5.6. Receiving a response [to a FloorQuery Message] (12.1.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a FloorQuery from a participant, the floor control server MUST respond with a FloorStatus message within the transaction failure window to complete the transaction.

5.7. Receiving a response [to a FloorRequestQuery Message] (12.2.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a FloorRequestQuery from a participant, the floor control server MUST respond with a FloorRequestStatus message within the transaction failure window to complete the transaction.

5.8. Receiving a response [to a UserQuery Message] (12.3.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a UserQuery from a participant, the floor control server MUST respond with a UserStatus message within the transaction failure window to complete the transaction.

5.9. Receiving a response [to a Hello Message] (12.4.2)

Prepend the sentence below at the start of this subsection:

> When communicating over unreliable transport and upon receiving a Hello from a participant, the floor control server MUST respond with a HelloAck message within the transaction failure window to complete the transaction.

5.10. Reception of a FloorRequestStatus Message (13.1.3)

The sentence below shall appear as a new subsection:

> When communicating over unreliable transport and upon receiving a FloorRequestStatus message from a floor control server, the participant MUST respond with a FloorRequestStatusAck message within the transaction failure window to complete the transaction.

5.11. Reception of a FloorStatus Message (13.5.3)

The sentence below shall appear as a new subsection:

> When communicating over unreliable transport and upon receiving a FloorStatus message from a floor control server, the participant MUST respond with a FloorStatusAck message within the transaction failure window to complete the transaction.

5.12. Reception of an Error Message (13.8.1)

The sentence below shall appear as a new subsection:

> When communicating over unreliable transport and upon receiving an Error message from a floor control server, the participant MUST respond with a ErrorAck message within the transaction failure window to complete the transaction.

5.13. Security Considerations (14)

It is a requirement that the extension of BFCP for unreliable transports shall not introduce any new threats.

Note: work is currently underway investigating the adoption of DTLS as an appropriate transport mechanism for BFCP.

5.14. IANA Considerations - Primitive Subregistry (15.2)

This section instructs the IANA to register the following new values for the BFCP primitive subregistry.

Value	Primitive	Reference
14	FloorRequestStatusAck	RFC[XXXX]
15	ErrorAck	RFC[XXXX]
16	FloorStatusAck	RFC[XXXX]
17	Goodbye	RFC[XXXX]
18	GoodbyeAck	RFC[XXXX]

Table 4: BFCP primitive subregistry

5.15. IANA Considerations - Error Code Subregistry (15.4)

This section instructs the IANA to register the following new values for the BFCP Error Code subregistry.

Value	Meaning	Reference
10	Unable to parse message	RFC[XXXX]

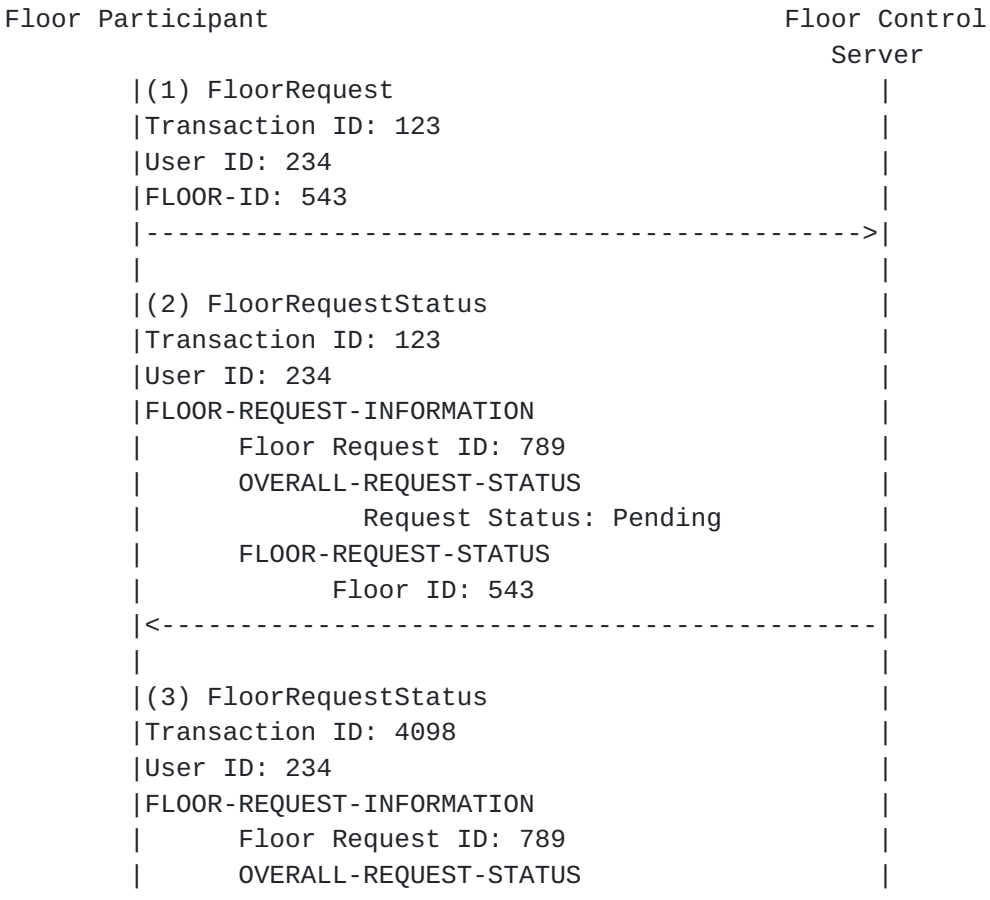
Table 5: BFCP Error Code subregistry

5.16. Example call flows for BFCP over Unreliable Transports (Appendix A)

(Note: This is a new appendix to [RFC4582].)

With reference to Section 4.1, the following figures show representative call-flows for requesting and releasing a floor, and obtaining status information about a floor when BFCP is deployed over an unreliable transport. The figures here show a loss-less interaction.

Note: A future version of this draft will show an example with lost packets due to unreliable transport.



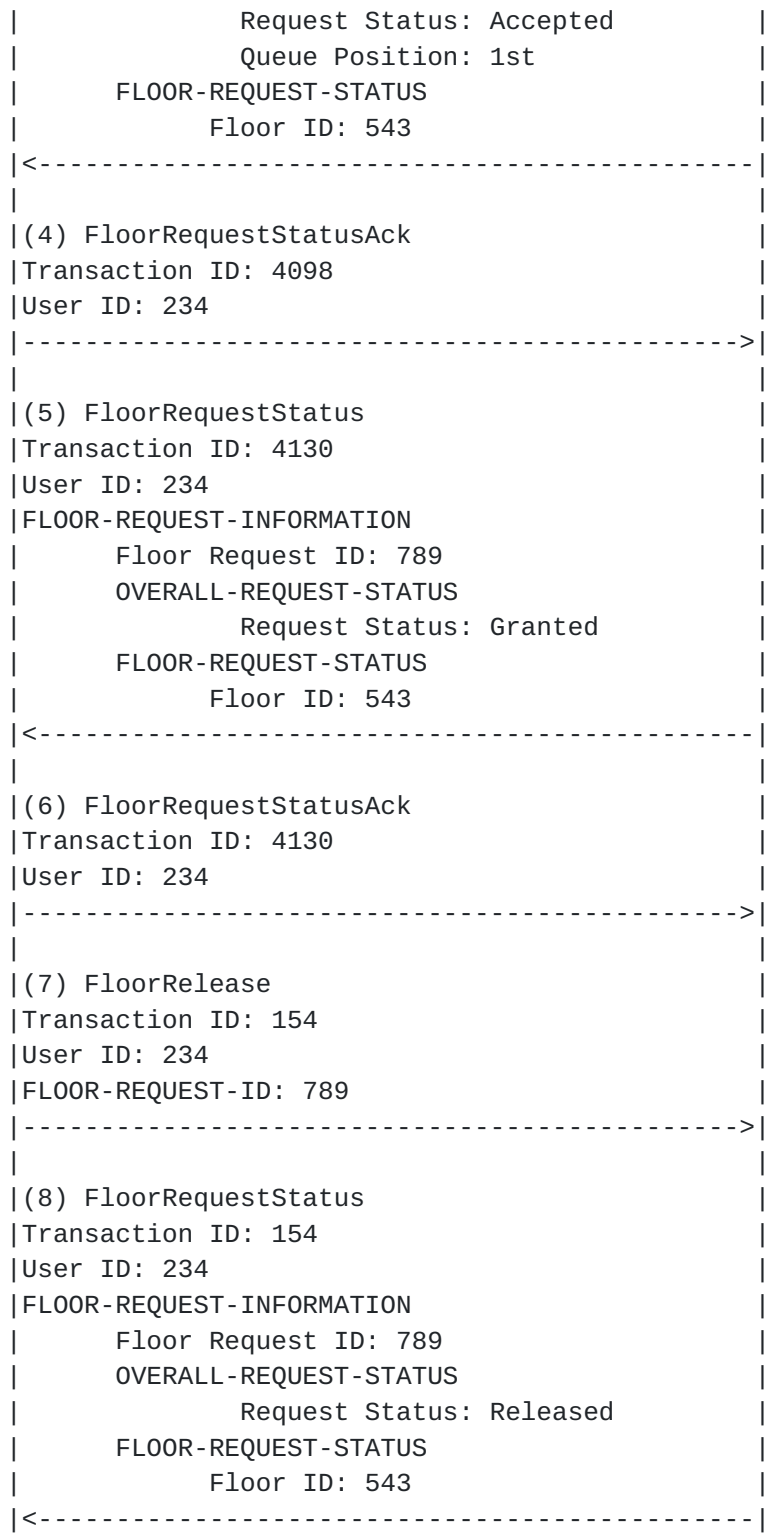
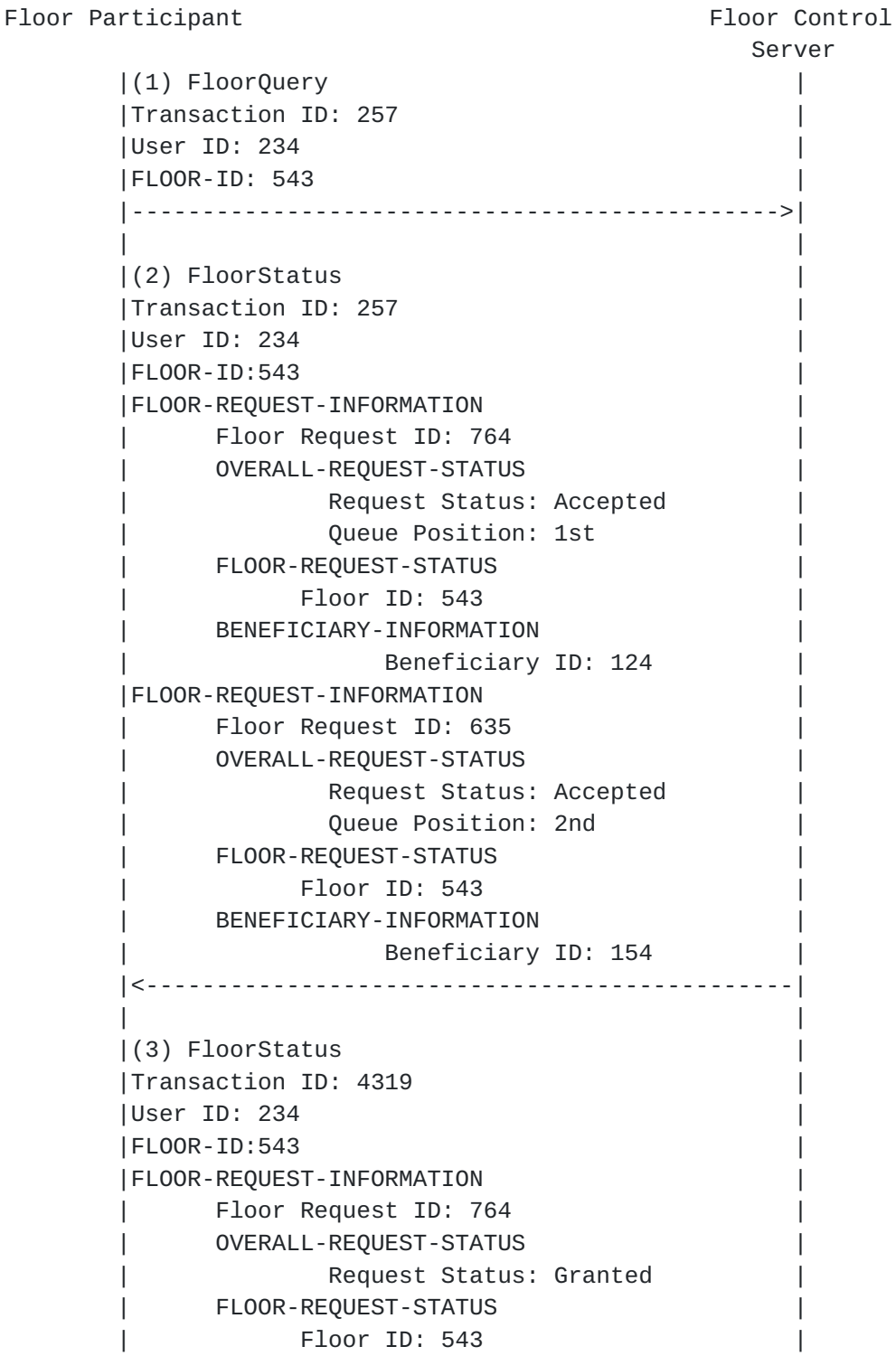


Figure 7: Requesting and releasing a floor

Note that in Figure 7, the FloorRequestStatus message from the floor control server to the floor participant is a transaction-closing

message as a response to the client-initiated transaction with Transaction ID 154. It does not and SHOULD NOT be followed by a FloorRequestStatusAck message from the floor participant to the floor control server.



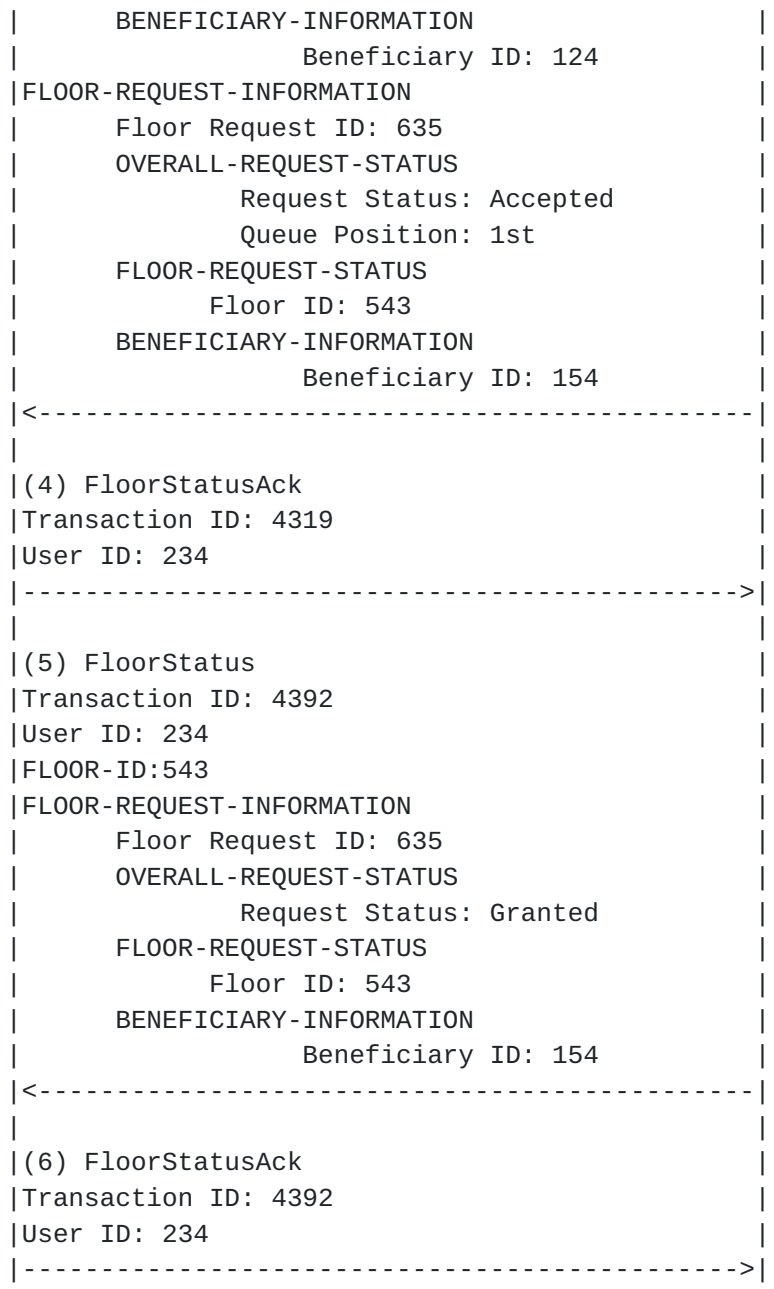


Figure 8: Obtaining status information about a floor

6. Revision of RFC4583

This section details revisions to [RFC4583], the format for specifying BFCP streams. The section number to which updates apply are indicated in parentheses in the titles of the sub-sections below.

6.1. Fields in the 'm' Line (3)

The section shall be re-written to remove reference to the exclusivity of TCP as a transport for BFCP streams.

1. In paragraph four, "... will initiate its TCP connection ..." becomes "... will direct BFCP messages ..."

2. In paragraph four, delete "Since BFCP only runs on top of TCP, the port is always a TCP port."

3. In paragraph five, we now define three new values for the transport field, adding "UDP/BFCP" as the third symbol, changing "former" for "first", "latter" for "second", and adding a final clause defining the use of UDP/BFCP as being for when BFCP runs on top of UDP

6.2. Security Considerations (10)

At this time, see Section 7.

6.3. Registration of SDP 'proto' values (11.1)

This section should be renamed now that there are more values to register in the SDP parameters registry, with the following added to the table:

Value	Reference
UDP/BFCP	RFC[XXXX]

Table 6: Value for the SDP 'proto' field

7. Future work

This draft reflects a work in progress, with at least the following items to be documented and/or revised before soliciting adoption by the XCON working group:

Secured transport Initial investigation has highlighted that the previously recommended approach of re-using Hello and HelloAck messages to open and maintain NAT pinholes is inadequate when considering the adoption of DTLS as a transport security mechanism. However, at this time insufficient work has been done to confirm DTLS as a recommendation, particularly as

regards signaling DTLS roles, key exchange, etc. It is likely that [I-D.ietf-sip-dtls-srtp-framework] will inform this investigation.

Protocol revision Certain aspects of this draft require different behaviors depending on whether a reliable or unreliable transport is being used, e.g. server-initiated transactions having Transaction ID 0 over reliable transports without acknowledgements versus non-zero and active-unique with an acknowledgement message when entities communicate over unreliable transports. If we allow TCP-based implementations to follow the graceful-close behaviour of [RFC4582] without mandating that the Goodbye message be signaled then it would not be necessary to bump the protocol version number. TCP-based implementations could continue as-is, whilst UDP-based implementations would be at their first version and as such no backward compatible issues would be present.

Fragmentation It has been observed that BFCP message structures can grow to be sufficiently large that they exceed the typical MTU threshold for local area networks (assumed here as 1500 octets). For example, a FloorStatus message with multiple FLOOR-REQUEST-INFORMATION attributes that contain detailed STATUS-INFO in the OVERALL-REQUEST-STATUS and FLOOR-REQUEST-STATUS attributes. A strategy for coping with such fragmented messages is required. Currently, this is held with a broad-sweeping statement of intent that implementations should restrict the size of their messages. Further refinement is likely required, such as an applicability statement on those BFCP messages and/or attributes deemed as inappropriate for use over transports where fragmentation is a concern, or further protocol specification to eradicate fragmentation as an issue.

Example signaling flows The next revision of this draft will include further example signaling exchanges over unreliable transport showing updated transactions and message retransmission as a visual aid and reference for implementors.

8. Acknowledgements

The team working on this draft are: Trond G. Andersen, Tom Kristensen, Eoin McLeod, Geir A. Sandbakken and Mark K. Thompson at TANDBERG; Alfred E. Heggestad at Telio Telecom.

9. Normative References

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[I-D.ietf-mmusic-ice-tcp]
Perreault, S. and J. Rosenberg, "TCP Candidates with Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-tcp-08 (work in progress), October 2009.

[I-D.ietf-sip-dtls-srtp-framework]
Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing an SRTP Security Context using DTLS", draft-ietf-sip-dtls-srtp-framework-07 (work in progress), March 2009.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4582] Camarillo, G., Ott, J., and K. Drage, "The Binary Floor Control Protocol (BFCP)", RFC 4582, November 2006.

[RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams", RFC 4583, November 2006.

Appendix A. Changes to previous drafts

A.1. -01 to -02

1. Stepped away from changing semantics and directionality of Hello and HelloAck messages for pinhole establishment and keep-alive in favour of ICE toolset, particularly as this would have not resolved connectivity establishment as a precursor to deployment of DTLS [RFC4347] as a transport security mechansim.

2. Change to COMMON-HEADER to reserve bit-16 of Transaction ID to show originator of transaction such that request/response and response/acknowledgement mapping can be maintained without colliding randomly chosen Transaction IDs. This also avoids a three-way handshake scenario around FloorRequest where the implicit acknowledgement (in FloorRequestStatus) might also be interpreted as a transaction openening request on the part of the floor control server.

3. Defined additional timer (T2) to soak up lost responses without additional processing.

4. Restricted outstanding transactions to only one in-flight per direction at any one time to mitigate re-ordering issues.

5. Defined entity behaviour when transactions timeout.

6. Specified initial suggestion for how to minimise fragmentation of messages.

7. Removed consideration of TCP-over-UDP after internal review.

8. Re-stated DTLS as likely preferred mechanism of securing transport, although this investigation is on-going.

A.2. -00 to -01

1. Refactored to a format that represents explicit changes to base RFCs.

2. Introduction of issues currently under investigation that preclude adoption.

3. Specified retransmission timer for requests.

Authors' Addresses

Mark K. Thompson (editor)
TANDBERG
Philip Pedersens vei 22
N-1366 Lysaker
Norway

Phone: +44-118-934-8711
Email: mark.thompson@tandberg.com
URI: http://www.tandberg.com/

Tom Kristensen
TANDBERG

Email: tom.kristensen@tandberg.com, tomkri@ifi.uio.no

Geir A. Sandbakken
TANDBERG

Email: geir.sandbakken@tandberg.com

Trond G. Andersen
TANDBERG

Email: trond.andersen@tandberg.com

Eoin McLeod
TANDBERG

Email: eoin.mcleod@tandberg.com