BFCPbis Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2012

T. Kristensen, Ed.
C. Eckel
A. Heggestad
G. Sandbakken
Cisco

February 17, 2012

Revision of the Binary Floor Control Protocol (BFCP) for use over an unreliable transport
draft-ietf-bfcpbis-rfc4582bis-01

Abstract

This draft describes how to extend the Binary Floor Control Protocol (BFCP) for use over an unreliable transport. It details the differences from the BFCP protocol definition document and the Session Description Protocol (SDP) format specified for BFCP streams.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 20, 2012.

Copyright Notice

Table of Contents

1. Introduction

This draft describes how to extend 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.

This extension does not change the semantics of BFCP. It permits UDP as an alternate transport. Existing implementations, in the spirit of the approach detailed in earlier versions of this draft (see Appendix A), have demonstrated the approach to be feasible. Initial compatibility among implementations has been achieved at previous interoperability events. The purpose of this draft is to formalize and publish the extension from the standard specification to facilitate complete interoperability between implementations.

The content of this draft relates to the BFCP protocol specification [RFC4582] and the SDP format for describing BFCP streams [RFC4583]. This draft is written with the goal of identifying the extensions associated with adding support for UDP as an alternate transport to an existing BFCP implementation.

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. Motivation

In existing video conferencing deployments, BFCP is used to manage the floor for the content sharing associated with the conference. For peer to peer scenarios, including business to business conferences and point to point conferences in general, it is frequently the case that one or both endpoints exists behind a NAT/ firewall. BFCP roles are negotiated in the offer/answer exchange as specified in [RFC4583], resulting in one endpoint being responsible for opening the TCP connection used for the BFCP communication.

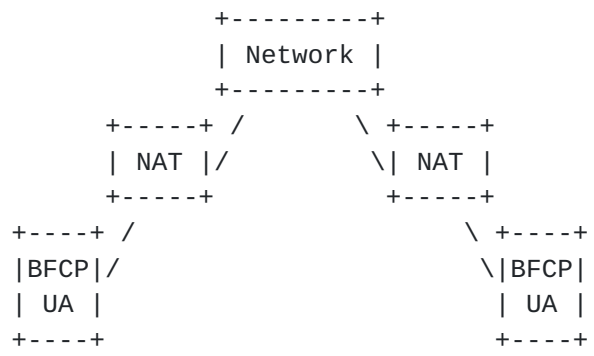


Figure 1: Use Case

The communication session between the video conferencing endpoints typically consists of a number of RTP over UDP media streams, for audio and video, and a BFCP connection for floor control. Existing deployments are most common in, but not limited to, enterprise networks. In existing deployments, NAT/firewall traversal for the RTP streams works using ICE and/or other methods, including those described in [I-D.ietf-mmusic-media-path-middleboxes].

When enhancing an existing SIP based video conferencing deployment with support for content sharing, the BFCP connection often poses a problem. The reasons for this fall into two general classes. First, there may be a strong preference for UDP based signaling in general. On high capacity endpoints (e.g. PSTN gateways or SIP/H.323 inter-working gateways), TCP can suffer from head of line blocking, and it uses many kernel buffers. Network operators view UDP as a way to avoid both of these. Second, establishment and traversal of the TCP connection involving ephemeral ports, as is typically the case with BFCP over TCP, can be problematic, as described in Appendix A of [I-D.ietf-mmusic-ice-tcp]. A broad study of NAT behavior and peer-to-peer TCP establishment for a comprehensive set of TCP NAT traversal techniques over a wide range of commercial NAT products concluded it was not possible to establish a TCP connection in 11% of the cases [IMC05]. The results are worse when focusing on enterprise NATs. A study of hole punching as a NAT traversal technique across a wide variety of deployed NATs reported consistently higher success rates when using UDP than when using TCP [P2PNAT].

To overcome the problems with establishing TCP flows between BFCP entities, this draft defines UDP as an alternate transport for BFCP, leveraging the same mechanisms in place for the RTP over UDP media streams for the BFCP communication. When using UDP as the transport, it is RECOMMENDED to follow the guidelines provided in [RFC5405]. NAT traversal for BFCP over UDP entities is discussed in more detail in Section 6.

The authors view this extension as a pragmatic solution to an existing deployment challenge.

3.1. Alternatives Considered

In selecting the approach of defining UDP as an alternate transport for BFCP, several alternatives were considered and explored to some degree. Each of these is discussed briefly in the following subsections. In summary, while these alternatives work in a number of scenarios, they are not sufficient, in and of themselves, to address the use case targeted by this draft.

3.1.1. ICE TCP

ICE TCP [I-D.ietf-mmusic-ice-tcp] extends ICE to TCP based media, including the ability to offer a mix of TCP and UDP based candidates for a single stream. ICE TCP has, in general, a lower success probability for enabling TCP connectivity without a relay if both of the hosts are behind a NAT (see Appendix A of [I-D.ietf-mmusic-ice-tcp]) than enabling UDP connectivity in the same scenarios. The happens because many of the currently deployed NATs in video conferencing networks do not support the flow of TCP hand shake packets seen in case of TCP simultaneous-open, either because they do not allow incoming TCP SYN packets from an address to which a SYN packet has been sent to recently, or because they do not properly process the subsequent SYNACK. Implementing various techniques advocated for candidate collection in [I-D.ietf-mmusic-ice-tcp] should increase the success probability, but many of these techniques require support from some network elements (e.g., from the NATs). Such support is not common in enterprise firewalls and NATs.

3.1.2. Teredo

Teredo [RFC4380] enables nodes located behind one or more IPv4 NATs to obtain IPv6 connectivity by tunneling packets over UDP. Teredo extensions [RFC6081] provide additional capabilities to Teredo, including support for more types of NATs and support for more efficient communication.

As defined, Teredo could be used to make BFCP work for the video conferencing use cases addressed in this draft. However, running the service requires the help of "Teredo servers" and "Teredo relays" [RFC4380]. These servers and relays generally do not exist in the existing video conferencing deployments. It also requires IPv6 awareness on the endpoints. It should also be noted that ICMP6, as used with Teredo to complete an initial protocol exchange and confirm that the appropriate NAT bindings have been set up, is not a conventional feature of IPv4 or even IPv6, and some currently

deployed IPv6 firewalls discard ICMP messages. As these networks continue to evolve and tackle the transaction to IPv6, Teredo servers and relays may be deployed, making Teredo available as a suitable alternative to BFCP over UDP.

3.1.3. GUT

GUT [I-D.manner-tsvwg-gut] attempts to facilitate tunneling over UDP by encapsulating the native transport protocol and its payload (in general the whole IP payload) within a UDP packet destined to the well-known port GUT_P. Unfortunately, it requires user-space TCP, for which there is not a readily available implementation, and creating one is a large project in itself. This draft has expired and its future is still not clear as it has not yet been adopted by a working group.

3.1.4. UPnP IGD

Universal Plug and Play Internet Gateway Devices (UPnP IGD) sit on the edge of the network, providing connectivity to the Internet for computers internal to the LAN, but do not allow Internet devices to connect to computers on the internal LAN. IGDs enable a computer on an internal LAN to create port mappings on their NAT, through which hosts on the Internet can send data that will be forwarded to the computer on the internal LAN. IGDs may be self-contained hardware devices or may be software components provided within an operating system.

In considering UPnP IGD, several issues exist. Not all NATs support UPnP, and many that do support it are configured with it turned off by default. NATs are often multilayered, and UPnP does not work well with such NATs. For example, a typical DSL modems acts as a NAT, and the user plugs in a wireless access point behind that, which adds another layer NAT. The client can discover the first layer of NAT using multicast but it is harder to figure out how to discover and control NATs in the next layer up.

3.1.5. NAT PMP

The NAT Port Mapping Protocol (NAT PMP) allows a computer in a private network (behind a NAT router) to automatically configure the router to allow parties outside the private network to contact it. NAT PMP runs over UDP. It essentially automates the process of port forwarding. Included in the protocol is a method for retrieving the public IP address of a NAT gateway, thus allowing a client to make this public IP address and port number known to peers that may wish to communicate with it.

Many NATs do not support PMP. In those that do support it, it has similar issues with negotiation of multilayer NATs as UPnP. Video conferencing is used extensively in enterprise networks, and NAT PMP is not generally available in enterprise-class routers.

4. Difference from RFC4582

This section details the difference from [RFC4582], the base protocol specification of BFCP, required for use over an unreliable transport. The section numbers to which differences apply are indicated in parentheses in the titles of the sub-sections below.

4.1. 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.

4.1.1. 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 (Editorial Note: here-in Section 4.31) shows the same sample interactions but over an unreliable transport.

4.2. COMMON-HEADER Format (5.1)

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

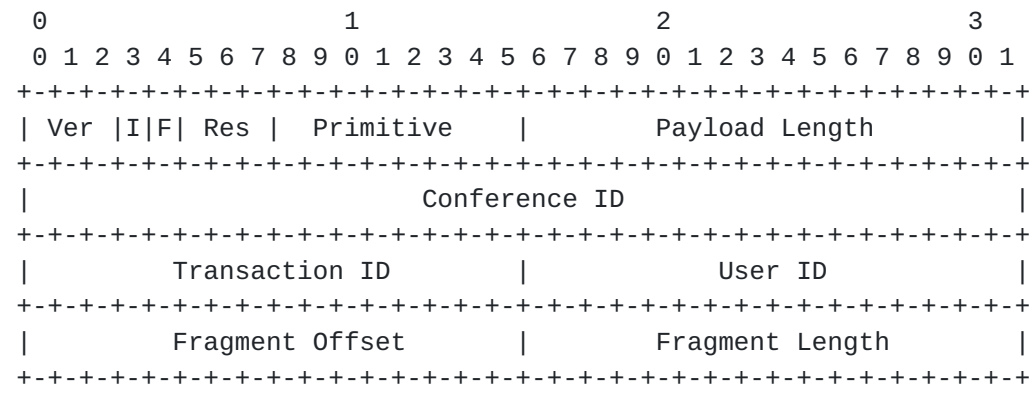


Figure 2: COMMON-HEADER format

The description for "Ver" is changed as follows on page 15:

> Ver: The 3-bit version field MUST be set to 1 when using BFCP over reliable transport, i.e. as in [RFC4582]. The 3-bit version field MUST be set to 2 when using BFCP over unreliable transport, with the extensions specified in this document.

The following text precedes "Reserved" on page 15:

> I: The Transaction Initiator (I) flag-bit has relevance only for use of BFCP over unreliable transport. When cleared, it indicates that this message is a request initiating a new transaction, and the Transaction ID that follows has been generated for this transaction. When set, it indicates that this message is a response to a previous request, and the Transaction ID that follows is the one associated with that request. When BFCP is used over reliable transports, the flag has no significance and SHOULD be cleared.
>
> F: The Fragmentation (F) flag-bit has relevance only for use of BFCP over unreliable transport. When cleared, the message is not fragmented. When set, it indicates that the message is a fragment of a large fragmented BFCP message. (The optional fields Fragment Offset and Fragment Length described below are present only if the F flag is set).

The Reserved field changes name to Res due to limited space in the ASCII graphic in Figure 2. In the description of the Reserved field "the 5 bits" is changed to "the 3 bits".

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 MUST 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 ; P <- S ; Ch <- S
18	GoodbyeAck	P -> S ; Ch -> S ; P <- S ; Ch <- S

Table 1: BFCP primitives

The following text should be added after "User ID" on page 15:

Fragment Offset: This optional field is present only if the F flag is set and contains a 16-bit value that specifies the number of 4-octet units contained in previous fragments.

Fragment Length: This optional field is present only if the F flag is set and contains a 16-bit value that specifies the number of 4-octet units contained in this fragment.

4.3. 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
11	Use DTLS

Table 2: Error Code meaning

4.4. FloorRequestStatusAck (5.3.14)

This new subsection specifies 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 3: FloorRequestStatusAck format

4.5. ErrorAck (5.3.15)

This new subsection specifies 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 4: ErrorAck format

4.6. FloorStatusAck (5.3.16)

This new subsection specifies 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 5: FloorStatusAck format

4.7. Goodbye (5.3.17)

This new subsection specifies 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 6: Goodbye format

4.8. GoodbyeAck (5.3.18)

This new subsection specifies 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 7: GoodbyeAck format

4.9. Transport (6)

An additional behavior is recommended for entities participating in communication over an unreliable transport that either wish to leave or are asked to leave an established BFCP connection, as detailed in the revised section introduction text below.

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 offer/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.

4.9.1. 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 compatibility 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.

4.9.2. Unreliable Transport (6.2)

BFCP entities may elect to exchange BFCP messages using UDP datagrams. UDP is an unreliable transport where neither delivery nor ordering is assured. Each BFCP UDP datagram MUST contain exactly one BFCP message. In the event the size of a BFCP message exceeds the MTU size, the BFCP message will be fragmented at the IP layer. Considerations related to fragmentation are covered in Section 4.9.3. 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 transmission 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 characterized 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. [Editorial note: would it be more straightforward to have all FloorRequestStatus messages acknowledged with a

FloorRequestStatusAck message?]

By restricting entities to having at most one pending transaction open, both the out-of-order receipt of messages as well as the possibility for congestion are mitigated. Additional details regarding congestion control are provided in Section 4.9.2.1. A server-initiated request (e.g. a FloorStatus 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 MUST be treated as superseding the information conveyed in any 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.

4.9.2.1. Congestion Control

BFCP may be characterized to generate "low data-volume" traffic, per the classification in [RFC5405]. Nevertheless is it necessary to ensure suitable and necessary congestion control mechanisms are used for BFCP over UDP. As described in previous paragraph every entity - client or server - is only allowed to send one request at a time, and await the acknowledging response. This way at most one datagram is sent per RTT given the message is not lost during transmission. In case the message is lost, the request retransmission timer T1 specified in Section 4.14 will fire and the message is retransmitted up to three times. The default initial interval is set to 500ms and the interval is doubled after each retransmission attempt, this is identical to the specification of the T1 timer in SIP as described in Section 17.1.1.2 of [RFC3261].

4.9.2.2. ICMP Error Handling

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, one

possibility is to utilize the peer connectivity checks, relay use and NAT pinhole maintenance mechanisms defined in ICE [RFC5245].

4.9.3. Large Message Considerations

Large messages become a concern when using BFCP if the overall size of a single BFCP message exceeds that representable within the 16-bit Payload Length field of the COMMON-HEADER. When using UDP, there is the added concern that a single BFCP message can be fragmented at the IP layer if its overall size exceeds the MTU threshold of the network.

4.9.3.1. Fragmentation Handling

When transmitting a BFCP message with size greater than the MTU, the sender should fragment the message into a series of N contiguous data ranges. The sender should then create N BFCP fragment messages (one for each data range) with the same Transaction ID. The size of each of these N messages MUST be smaller than the MTU. The F flag in the COMMON-HEADER is set to indicate fragmentation of the BFCP message.

For each of these fragments the Fragment Offset and Fragment Length fields are included in the COMMON-HEADER. The Fragment Offset field denotes the number of bytes contained in the previous fragments. The Fragment Length contains the length of the fragment itself. Note that the Payload Length field contains the length of the entire, unfragmented message.

When a BFCP implementation receives a BFCP message fragment, it MUST buffer the fragment until it has received the entire BFCP message. The state machine should handle the BFCP message only after all the fragments for the message have been received.

If a fragment of a BFCP message is lost, the sender will not receive an ACK for the message. Therefore the sender will retransmit the message with same transaction ID as specified in Section 4.13. If the ACK sent by the receiver is lost, then the entire message will be resent by the sender. The receiver MUST then retransmit the ACK. The receiver can discard an incomplete buffer utilizing the Response Retransmission Timer, starting the timer after the receipt of the first fragment.

4.10. Lower-Layer Security (7)

Expand the section to mandate support for DTLS when transport over UDP is used such that it reads as follows:

BFCP relies on lower-layer security mechanisms to provide replay and integrity protection and confidentiality. BFCP floor control servers and clients (which include both floor participants and floor chairs) MUST support TLS for transport over TCP and MUST support DTLS for transport over UDP [RFC5246]. Any BFCP entity MAY support other security mechanisms.

BFCP entities MUST support, at a minimum, the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite [RFC5246].

Which party, the client or the floor control server, acts as the TLS/DTLS server depends on how the underlying TLS/DTLS connection is established. For a TCP/TLS connection established using an SDP offer/answer exchange [RFC4583], the answerer (which may be the client or the floor control server) always acts as the TLS server. For a UDP/DTLS connection established using the same exchange, either party can be the DTLS server depending on the setup attributes exchanged, as defined in [RFC5763].

4.11. Protocol Transactions (8)

The final clause of the introduction to section 8 should be read as:

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 4.13) until the transaction is completed.

4.12. Server Behavior (8.2)

The final clause of this section should be read as:

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.

4.13. 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.

4.14. 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 response 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.

4.15. 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.

4.16. 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

The default value for T1 is 500 ms, this is an estimate of the RTT for completing the transaction. T1 MAY be chosen larger, and this is RECOMMENDED if it is known in advance that the RTT is larger. Regardless of the value of T1, the exponential backoffs on retransmissions described in Section 4.14 MUST be used.

4.17. Authentication and Authorization (9)

The first sentence of the second paragraph should be read as:

> BFCP supports TLS/DTLS mutual authentication between client and floor control servers, as specified in section 9.1.

4.17.1. TLS Based Mutual Authentication (9.1)

Change each instance of "TLS" to "TLS/DTLS", and each instance of "TCP" to "TCP/UDP".

4.18. 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.

4.19. 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.

4.20. 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.

4.21. 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.

4.22. 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.

4.23. 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.

4.24. 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.

4.25. 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.

4.26. 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.

4.27. 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.

4.28. Security Considerations (14)

Change each instance of "TLS" to "TLS/DTLS", and each instance of "TCP" to "TCP/UDP".

4.29. 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 4582bis
15	ErrorAck	RFC 4582bis
16	FloorStatusAck	RFC 4582bis
17	Goodbye	RFC 4582bis
18	GoodbyeAck	RFC 4582bis

Table 4: BFCP primitive subregistry

4.30. IANA Considerations - Error Code Subregistry (15.4)

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

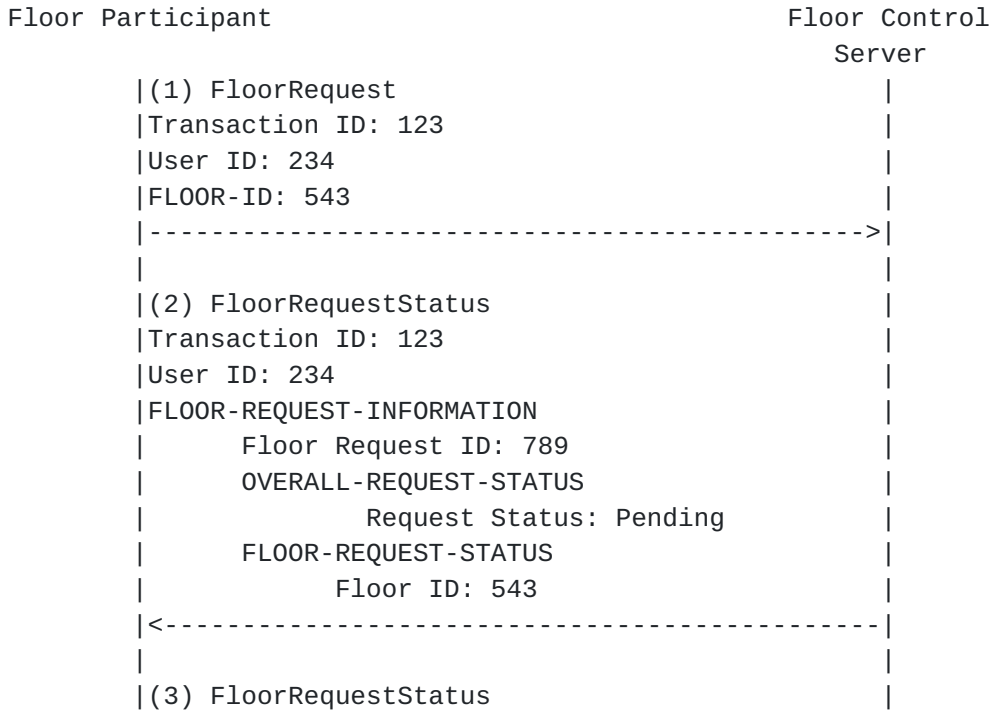
Value	Meaning	Reference
10	Unable to parse message	RFC 4582bis
11	Use DTLS	RFC 4582bis

Table 5: BFCP Error Code subregistry

4.31. Example Call Flows for BFCP over Unreliable Transport (Appendix A)

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.

Editorial Note: A future version of this draft will show an example with lost packets due to unreliable transport, as well as examples on usage of DTLS and STUN in call the setup phase.



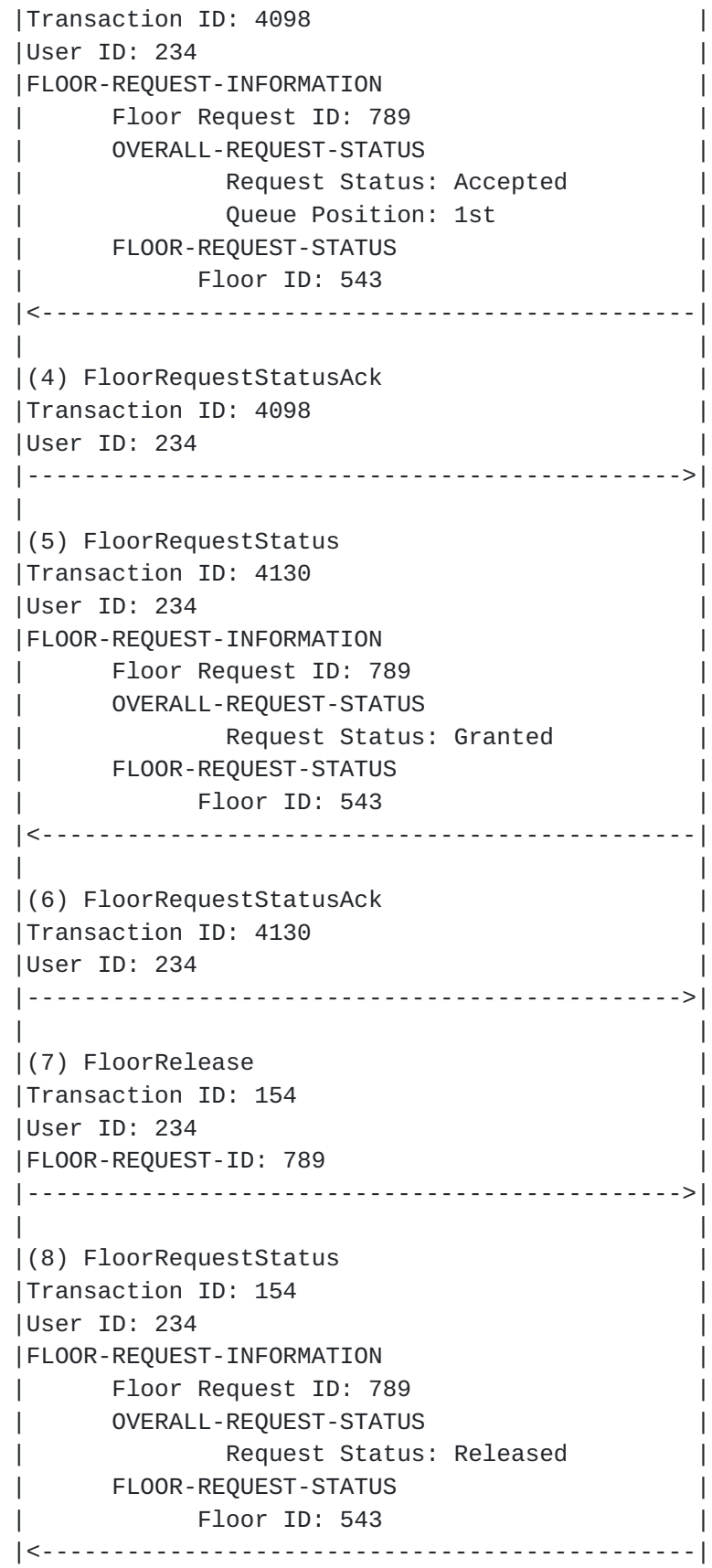
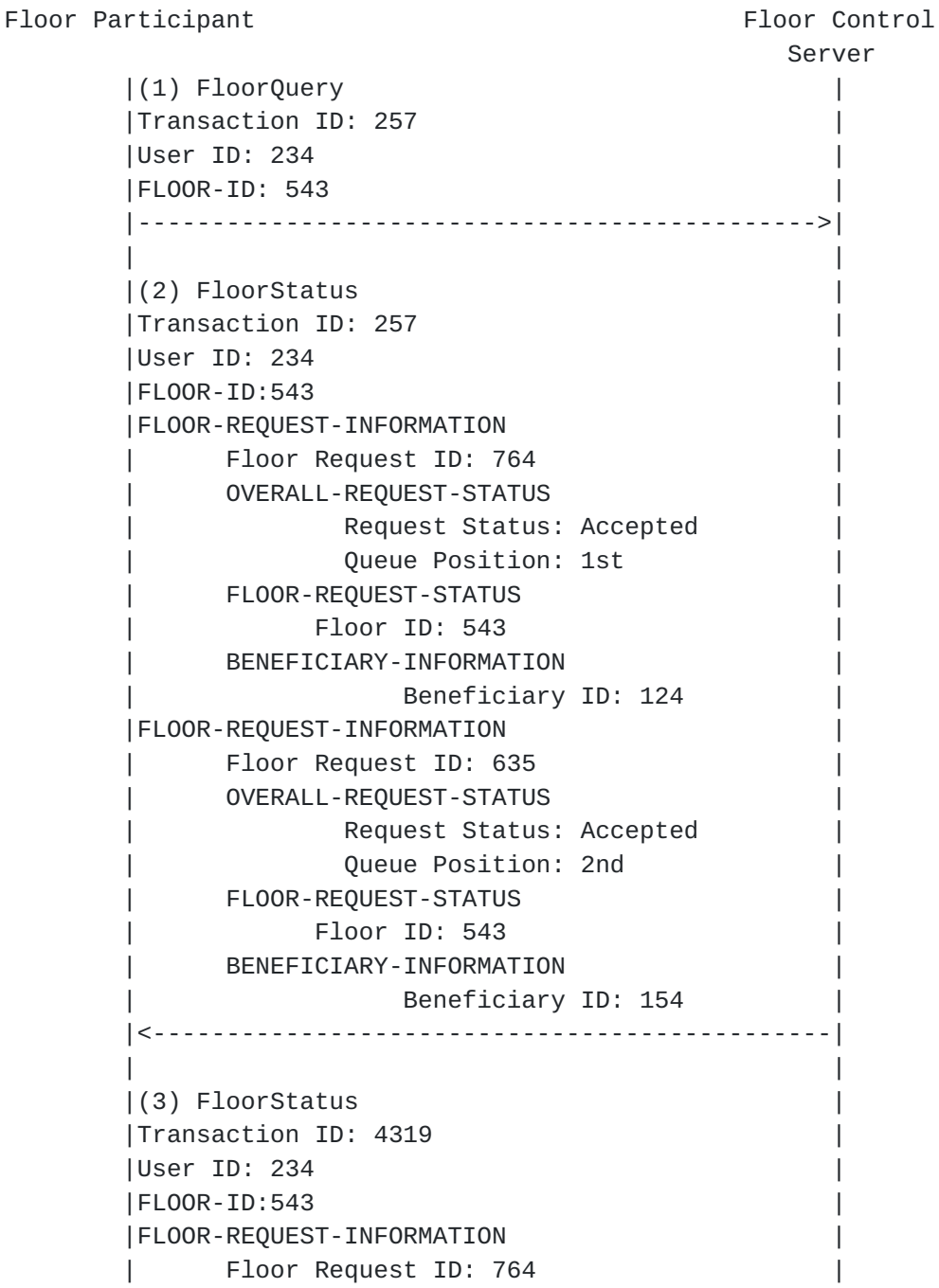

|Transaction ID: 4098
|User ID: 234
|FLOOR-REQUEST-INFORMATION
| Floor Request ID: 789
| OVERALL-REQUEST-STATUS
| Request Status: Accepted
| Queue Position: 1st
| FLOOR-REQUEST-STATUS
| Floor ID: 543
|(4) FloorRequestStatusAck
|Transaction ID: 4098
|User ID: 234
|(5) FloorRequestStatus
|Transaction ID: 4130
|User ID: 234
|FLOOR-REQUEST-INFORMATION
| Floor Request ID: 789
| OVERALL-REQUEST-STATUS
| Request Status: Granted
| FLOOR-REQUEST-STATUS
| Floor ID: 543
|(6) FloorRequestStatusAck
|Transaction ID: 4130
|User ID: 234
|(7) FloorRelease
|Transaction ID: 154
|User ID: 234
|FLOOR-REQUEST-ID: 789
|(8) FloorRequestStatus
|Transaction ID: 154
|User ID: 234
|FLOOR-REQUEST-INFORMATION
| Floor Request ID: 789
| OVERALL-REQUEST-STATUS
| Request Status: Released
| FLOOR-REQUEST-STATUS
| Floor ID: 543

Figure 8: Requesting and releasing a floor

Note that in Figure 8, 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.



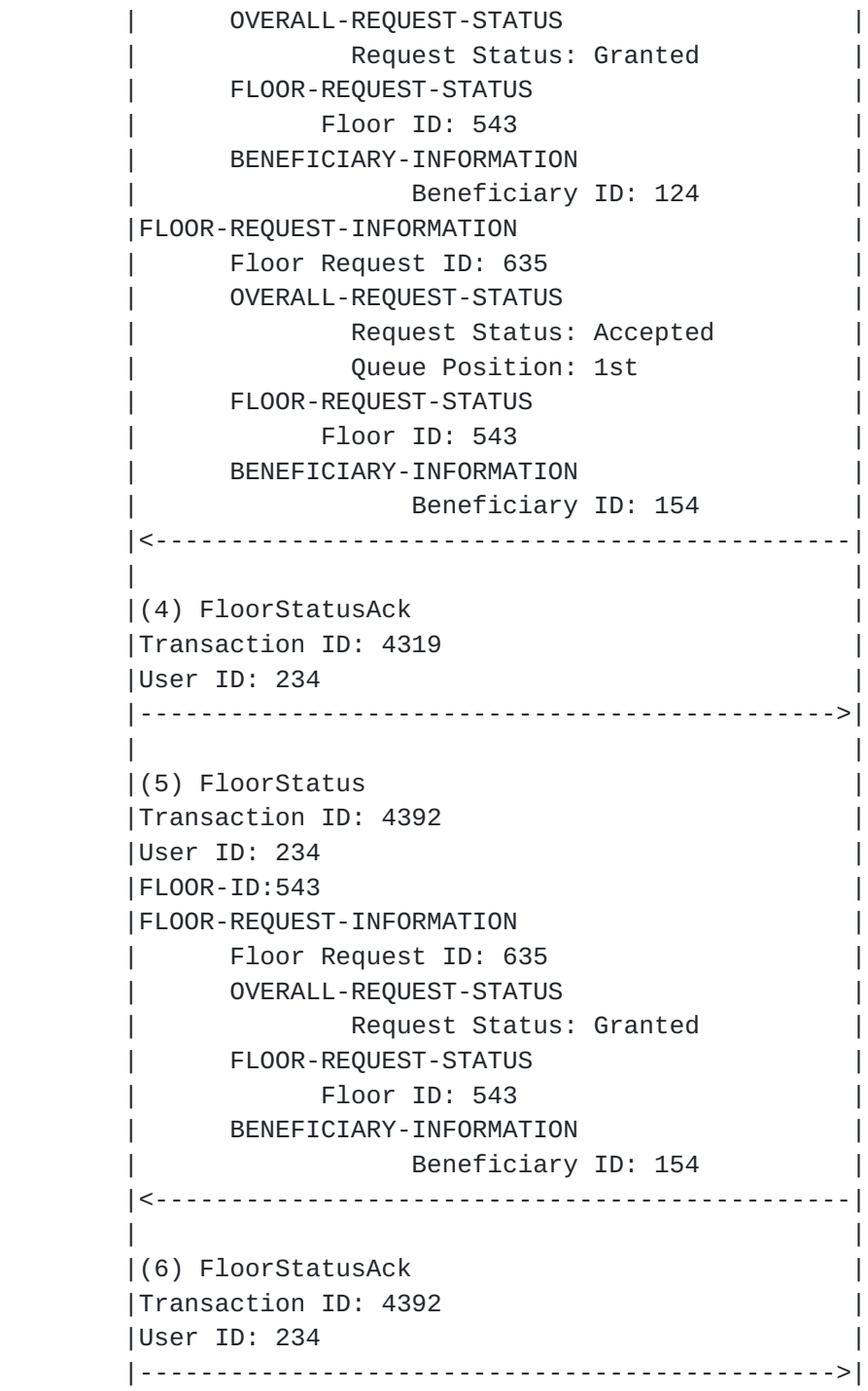


Figure 9: Obtaining status information about a floor

5. Revision of RFC4583

This section details revisions to [RFC4583], the SDP format for specifying BFCP streams. The section number to which updates apply

are indicated in parentheses in the titles of the sub-sections below.

5.1. 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. Change paragraph five, "We define two new values ... ", to, "We define four new values for the transport field: TCP/BFCP, TCP/TLS/BFCP, UDP/BFCP, and UDP/TLS/BFCP. TCP/BFCP is used when BFCP runs directly on top of TCP, and TCP/TLS/BFCP is used when BFCP runs on top of TLS, which in turn runs on top of TCP. Similarly, UDP/BFCP is used when BFCP runs directly on top of UDP, and UDP/TLS/BFCP is used when BFCP runs on top of DTLS [RFC4347], which in turn runs on top of UDP."

5.2. Authentication (8)

In last paragraph, change "When TLS is used, once the underlaying TCP connection is established" to "When TLS is used with TCP, once the underlying connection is established".

5.3. Security Considerations (10)

Append to the first paragraph, "Furthermore, when using DTLS over UDP, considerations for its use with RTP and RTCP are presented in [RFC5763]. The requirements for the offer/answer exchange, as listed in Section 5 of that document, MUST be followed."

5.4. 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 4583bis
UDP/TLS/BFCP	RFC 4583bis

Table 6: Value for the SDP 'proto' field

6. NAT Traversal

One of the key benefits when using UDP for BFCP communication is the ability to leverage the existing NAT traversal infrastructure and strategies deployed to facilitate transport of the media associated with the video conferencing sessions. Depending on the given deployment, this infrastructure typically includes some subset of ICE [RFC5245].

In order to facilitate the initial establishment of NAT bindings, and to maintain those bindings once established, BFCP over UDP entities are RECOMMENDED to use STUN [RFC5389] for keep-alives, as described for SIP [RFC5626]. This results in each BFCP entity sending a packet, both to open the pinhole and to learn what IP/port the NAT assigned for the binding.

In order to facilitate traversal of BFCP packets through NATs, BFCP over UDP entities are RECOMMENDED to use symmetric ports for sending and receiving BFCP packets, as recommended for RTP/RTCP [RFC4961].

7. Remaining Work

This draft reflects a work in progress, with at least the following items to be documented and/or revised:

Example signaling flows: A later version of this draft will include further examples - as appropriate - of signaling exchanges over unreliable transport as a visual aid and reference for implementers, potential candidates: Updated transactions, message retransmission, usage of DTLS during call setup, and combined usage of DTLS and STUN.

Reformat and merge: After figuring out the technical details in this draft, the "diff" will be merged to form proper bis-drafts to become RFC4582bis (in BFCPbis WG) and RFC4583bis (in MMUSIC WG).

Other issues not related to transport Fixing erratas to the RFCs and known minor issues with the existing specification.

8. Contributing Authors

The authors/editors would like to thank Mark K. Thompson, Eoin McLeod and Nivedita Melinkeri who made a major contribution to the development of this document.

Eoin McLeod
Cisco
Email: eoimcleo@cisco.com

Nivedita Melinkeri
Cisco
Email: nivedita@cisco.com

Mark K. Thompson
Cisco
Email: markth2@cisco.com

9. Acknowledgements

We acknowledge contributions to one or more previous versions of this draft from Trond G. Andersen, Gonzalo Camarillo, Roni Even, Lorenzo Miniero, Joerg Ott, Hadriel Kaplan, Dan Wing, Cullen Jennings, David Benham, and Alan Ford.

10. References

10.1. Normative References

[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.

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

10.2. Informative References

[I-D.ietf-mmusic-ice-tcp]
Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-tcp-16 (work in progress), November 2011.

[I-D.ietf-mmusic-media-path-middleboxes]
Stucker, B. and H. Tschofenig, "Analysis of Middlebox Interactions for Signaling Protocol Communication along the Media Path", draft-ietf-mmusic-media-path-middleboxes-03 (work in progress), July 2010.

[I-D.manner-tsvwg-gut]
Manner, J., Varis, N., and B. Briscoe, "Generic UDP Tunnelling (GUT)", draft-manner-tsvwg-gut-02 (work in progress), July 2010.

[IMC05] Guha, S. and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls", 2005, <http://saikat.guha.cc/pub/imc05-tcpnat.pdf/>.

[P2PNAT] Ford, B., Srisuresh, P., and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators", April 2005, <http://www.brynosaurus.com/pub/net/p2pnat.pdf/>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, May 2010.

[RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, January 2011.

Appendix A. Change History

A.1. draft-ietf-bfcpbis-rfc4582bis-00 to -01

1. Mandated using version 2 (Ver field == 2) for BFCP over UDP, with the extensions described in this draft. For BFCP over TCP, the version is still 1.

2. Added text regarding fragmentation handling: A new 'F' flag and Fragment Offset field in Section 4.2. Added fragmentation handling mechanism in Section 4.9.3.1.

3. Resolve an inconsistency between Section 4.10 and Section 5.3, by introducing the setup attribute for DTLS.

4. Moved some authors to the new Section 8 Contributing Authors.

5. A dash of editorial polish.

A.2. draft-sandbakken-dispatch-bfcp-udp-03 to draft-ietf-bfcpbis-rfc4582bis-00

1. Draft name change. Adopted as main work item in BFCPbis WG.

2. Switched from informational to standards track.

3. No conflict with IANA registries for BFCP, since the aim is a standards track RFC. Removed text in Future work section.

4. Just editorial changes as requested by WG chairs; used as a starting point in the new WG. Will add changes in upcoming version. Also author list will be considered, for instance adding a contributors section in the draft.

A.3. draft-sandbakken-dispatch-bfcp-udp-02 to -03

1. Added fragmentation and reassembly mechanism defined for RELOAD as a candidate mechanism for consideration for BFCP when transported over UDP.

2. Added ERROR-CODE to indicate DTLS is required.

3. Added UDP/TLS/BFCP as 4th transport value for BFCP.

4. Added requirement to follow offer/answer procedure in [RFC5763] when using DTLS over UDP for BFCP.

A.4. draft-sandbakken-dispatch-bfcp-udp-01 to -02

1. Switched from standards track to informational.

2. Added section on motivation, including alternatives considered, to address issues raised at IETF 79 and on various workgroup aliases.

3. Changed semantics of the Transaction Initiator (I) flag-bit.

4. Expanded transport section to more explicitly call out considerations regarding congestion control and ICMP errors, and add considerations for large messages.

5. Updated security related sections and added authentication section to address DTLS when using UDP.

6. Added section on NAT Traversal.

7. Some editorial changes.

A.5. draft-sandbakken-dispatch-bfcp-udp-00 to -01

1. Decision made to not increase the protocol version number as a result of this extension. 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. As the graceful-close behavior of [RFC4582] is still allowed for TCP-based implementations without mandating the use of the new Goodbye message, there is no need to change the version number.

2. Removed the - a bit too verbose - rationale/motivation text describing background and why other approaches where not chosen. Was OK for a -00 draft, not strictly needed.

3. Not mandate ICE as a SHALL, but leave it as a non-mandatory way of solving the potential need for NAT/FW traversal.

4. Emphasized that the reference to DTLS-SRTP are merely informational.

5. A dash of polish and nitpicking added, some typos fixed.

A.6. draft-sandbakken-xcon-bfcp-udp-02 to draft-sandbakken-dispatch-bfcp-udp-00

1. Draft name change. As XCON WG is closing this draft is submitted to Dispatch WG as the arena of discussion.

2. Moved Transaction Identifier bit (I) from the Transaction ID to one of the current 5 reserved bits. Keep current Transaction ID syntax and semantics. Avoid potential problems with existing TCP based implementations.

3. The way congestion control is taken care of is explained, with reference to [RFC5405]. One message per RTT. Backoff and normative behavior for timer T1 clarified.

4. Mandated support for DTLS in case unreliable transport (i.e. UDP) is implemented. Details and examples to be included. Model after [RFC5763], details on how to adapt the SRTP associated details to BFCP and whether a reference or copying the text across and changing is needed.

5. Added the Rationale and Scope section to position and explain the motivation for this draft more in detail.

6. A number of typos and editorial changes.

A.7. draft-sandbakken-xcon-bfcp-udp-01 to -02

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

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 opening 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 behavior when transactions timeout.

6. Specified initial suggestion for how to minimize 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.8. draft-sandbakken-xcon-bfcp-udp-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

Tom Kristensen (editor)
Cisco
Philip Pedersens vei 22
N-1366 Lysaker
Norway

Email: tomkrist@cisco.com, tomkri@ifi.uio.no

Charles Eckel
Cisco
170 West Tasman Drive
San Jose, CA 95134
United States

Email: eckelcu@cisco.com

Alfred E. Heggestad
Cisco
Philip Pedersens vei 22
N-1366 Lysaker
Norway

Email: aheggest@cisco.com

Geir A. Sandbakken
Cisco
Philip Pedersens vei 22
N-1366 Lysaker
Norway

Email: geirsand@cisco.com