Workgroup: Network Working Group
Internet-Draft: draft-pardue-quic-http-mcast-11
Published: 4 July 2022
Intended Status: Experimental
Expires: 5 January 2023
Authors: L. Pardue R. Bradbury
BBC Research & Development
S. Hurst
BBC Research & Development

Hypertext Transfer Protocol (HTTP) over multicast QUIC

Abstract

This document specifies a profile of the QUIC protocol and the HTTP/3 mapping that facilitates the transfer of HTTP resources over multicast IP using the QUIC transport as its framing and packetisation layer. Compatibility with the QUIC protocol's syntax and semantics is maintained as far as practical and additional features are specified where this is not possible.

Table of Contents

1. Introduction

The means to bulk transfer resources over multicast IP [RFC1112] using HTTP semantics presents an opportunity to more efficiently deliver services at scale, while leveraging the wealth of existing HTTP-related standards, tools and applications. Audio-visual segmented media, in particular, would benefit from this mode of transmission.

The carriage of HTTP over multicast IP may be satisfied using existing technologies, for example the Real-time Transport Protocol (RTP) [RFC3550], File Delivery over Unidirectional Transport (FLUTE) [RFC6726], and NACK-Oriented Reliable Multicast (NORM) [RFC5740]. However, such protocols typically require the translation or encapsulation of HTTP. This introduces concerns for providers of services, such as defining the translation, additional workload, complication of workflows, manageability issues, versioning issues, and so on.

In contrast, this document describes a simpler and more direct expression of HTTP semantics over multicast IP. HTTP over multicast QUIC is a profile of the QUIC protocol [RFC9000] (Section 4) and the HTTP/3 mapping [RFC9114] (Section 5). This includes the repurposing of certain QUIC packet fields and changes to some protocol procedures (e.g. prohibition of the usage of certain frame types) which, in turn, change the behavioural expectations of endpoints. However, the profile purposely limits the scope of change in order to preserve maximum syntactic and semantic compatibility with conventional QUIC. For the reader's convenience, the differences between this specification and conventional QUIC are summarised in Appendix C.

This profile prohibits the transmission of QUIC packets from receiver to sender via multicast IP. The use of side-channel or out-of-band feedback mechanisms is not prohibited by this specification, but is out of scope and these are not considered further by the present document.

Experience indicates that a generally available multicast deployment is difficult to achieve on the Internet notwithstanding the improvements that IPv6 [RFC8200] makes in this area. There is evidence that discretely referenced multicast "islands" can more pragmatically be deployed. Discovery of such islands by receivers, as they become available, is typically difficult, however. To address the problem, this document describes an HTTP-based discovery mechanism that uses HTTP Alternative Services [RFC7838] to advertise the existence of multicast QUIC sessions (Section 3). This provides the means for multicast-capable endpoints to learn about and make use of them in an opportunistic and user-imperceptible manner. This mechanism results in a common HTTP application layer for both the discovery and delivery of services across unicast and multicast networks. This provides support for users and devices accessing services over a heterogeneous network. This is a departure from conventional multicast discovery technologies such as SDP [RFC8866] and SAP [RFC2974].

The discovery mechanism also addresses some of the issues related to using QUIC over a unidirectional network association by replacing

connection establishment aspects that depend on a bidirectional transport.

The present document includes a number of optional features. It is anticipated that further specifications will define interoperability profiles suited to particular application domains.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all captials, as shown here.

This document uses the Augmented BNF defined in [RFC5234] and updated by [RFC7405] along with the "#rule" extension defined in Section 7 of [RFC7230]. The rules below are defined in [RFC5234], [RFC7230], and [RFC7234]:

*quoted-string = <quoted-string, see [RFC7230], Section 3.2.6>

*token = <token, see [RFC7230], Section 3.2.6>

*uri-host = <uri-host, see [RFC7230], Section 2.7>

1.2. Definitions

Definitions of terms that are used in this document:

*endpoint: A host capable of being a participant in a multicast QUIC session.

*multicast QUIC session: A logical unidirectional flow of metadata and data over multicast IP, framed according to this specification. The lifetime of a session is independent of any endpoint.

*participant: A sender or receiver that is taking part in a multicast QUIC session.

*sender: A participant sending multicast traffic according to this specification.

*receiver: A participant receiving multicast traffic according to this specification.

*session: See multicast QUIC session.

*session ID: The identifier for a multicast QUIC session.

*session parameter: Characteristic of a multicast QUIC session.

2. Multicast QUIC Sessions

A QUIC connection [RFC9000] carried over bidirectional unicast is defined as a conversation between two QUIC endpoints that multiplexes several logical streams within a single encryption context. This is a one-to-one relationship. Furthermore, QUIC connections achieve decoupling from the underlying network (IP and port) by means of a set of Connection IDs, with each endpoint generating these IDs and using them to identify the direction of flow. Use of a consistent connection identifier allows QUIC connections to survive changes to the network connectivity. The establishment of a QUIC connection relies upon an up-front, in-band exchange (and possible negotiation) of cryptographic and transport parameters (conveyed in QUIC handshake messages).

The mapping of HTTP semantics over the QUIC transport protocol [RFC9114] facilitates the transfer of hypermedia over QUIC connections. The establishment of a HTTP/3 connection relies upon all the requirements stipulated for the transport, as well as communication of HTTP-specific settings (conveyed in HTTP/3 SETTINGS frames). Such parameters may be required or optional and may be used by either endpoint to control the characteristics of connection usage.

This concept of a connection does not suit the carriage of HTTP/3 over unidirectional network associations such as multicast IP. In fact, there is no requirement for either endpoint (multicast sender or receiver) to be in existence in order for the other to start or join this one-sided conversation. The term "connection" is misleading in this context; therefore we introduce an alternative term "multicast QUIC session" or simply "session", which is defined as the logical entity describing the characteristics of the anticipated unidirectional flow of metadata and data. Such characteristics are expressed as "session parameters", described in Section 2.2. Advertisement of multicast QUIC sessions, specified in Section 3, allows for the senders and receivers to discover a session and to form multicast IP network associations that permit traffic flow.

2.1. Session States

The lifecycle of a multicast QUIC session is decoupled from the lifecycle of any particular endpoint. Multicast receivers or senders that take part in a session are called participants. The state of a session is influenced by the actions of participants. The loose coupling of participants means that they are unlikely to have a consistent shared view of the current state of a session. There is no requirement for a participant to know the session state and the

present document does not define a method to explicitly determine it. The definitions of session states provided below are intended to assist higher-level operational treatment of sessions:

- *Quiescent: the session has no participants and is ready to accept them.

- *Half-established: the session has a participant.

- *Fully-established: the session has a sender and at least one receiver participant.

- *Finished: the session has ended, and there are no participants.

Permitted states transitions are shown in Figure 1 below.

The transmission of QUIC packets is expected to occur only during the Half-Established and Fully-Established states.

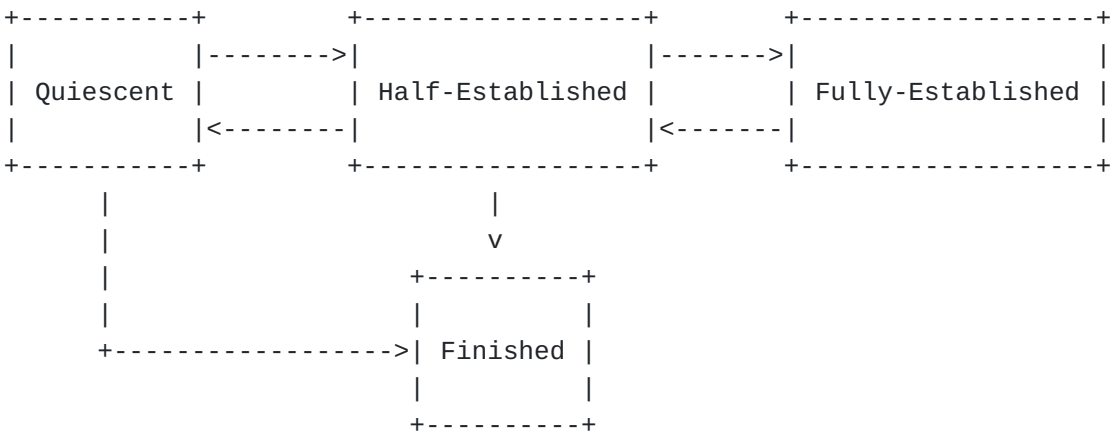

Figure 1: Multicast QUIC session states

2.1.1. Session Establishment

A session begins in the Quiescent state. A typical session establishment sequence would see the transition from Quiescent to Half-Established when a sender joins the session. The transition from Half-Established to Fully-Established occurs when at least one receiver joins the session.

It is equally valid for a receiver to join a session in the Quiescent state, triggering the transition to Half-Established. In this case, the transition to Fully-Established takes place only when a sender joins the session.

2.1.2. Session Termination

Participants MAY leave a session at any time. A session enters the Finished state when all participants have left it. Senders MAY signal their intent to leave using explicit session tear-down (Section 5.4). Receivers can detect that a sender has left via idle timeout (Section 3.3) and take that as a signal to leave, or receivers may leave as part of session migration (described in the next section).

In a typical case, a session that is in the Fully-Established state would be closed in two stages. In the first stage the sender sends explicit shutdown messages to the multicast group and subsequently stops transmitting packets. This causes the session to transition from Fully-Established to Half-Established. In the second stage, receivers that have received explicit shutdown messages leave the multicast group. Once all receivers have left the session it transitions from Half-Established to Finished.

The transition from Quiescent to Finished could also occur in response to out-of-band actions, for example the availability of a session being withdrawn without any participants having made use of it.

2.1.3. Session Migration

Endpoints MAY migrate between multicast QUIC sessions (for example, to make use of alternate session parameters that are preferred). Session migration requires participants to leave the current session and join the new session. These actions affect the state of the respective sessions as explained above.

The discovery of multicast QUIC sessions is described in Section 3.

2.2. Session Parameters

The characteristics of multicast QUIC sessions are expressed as session parameters, which cover cryptographic and transport parameters, HTTP-specific settings and multicast-specific configuration.

Session parameter exchange over IP multicast is difficult:

* In-band exchanges are one-way, and so the client does not have the means to send session parameters.

* The lifecycle of any multicast sender is independent of the multicast receiver. It is therefore unlikely that all receivers will have joined a session in time to receive parameters sent at the start of a multicast session.

A range of strategies exists to mitigate these points. However, each has the possibility to add complexity to deployment and manageability, transmission overhead, or other such concerns. This document defines a solution that relies on the one-way announcement of session parameters in advance of session establishment. This is achieved using HTTP Alternative Services [RFC7838] and is explained in Section 3. Other advertisement solutions are not prohibited but are not explored in this document.

Session parameters MUST NOT change during the lifetime of a session. This restriction also applies to HTTP-level settings (see Section 5.1).

2.3. Session Identification

This document defines an optional session identifier used to identify a session. This "Session ID" affords independence from multicast IP, creating the possibility for a session to span multiple multicast groups, or to migrate a session to a different multicast group. Assignment of Session ID is considered out of this document's scope.

The Session ID is carried in the Destination Connection ID field of the QUIC packet (see Section 4.3). Source Connection IDs are not used.

The maximum size of a Session ID is 160 bits. The size of the Destination Connection ID field used to convey the Session ID SHALL be the smallest number of full bytes required to represent the full Session ID value advertised in the session-id session parameter (Section 10.2.4). If no session-id parameter is advertised, then this session has a zero-length session ID, and the Destination Connection ID field SHALL be omitted from all QUIC packets related to the session.

A multicast sender participating in a session with an advertised session-id session parameter MUST send QUIC packets with a matching Session ID. Conversely, a multicast sender participating in a session without an advertised session-id session parameter MUST NOT send QUIC packets with a non-zero-length Destination Connection ID field.

A multicast receiver participating in a session with an advertised session-id session parameter MUST validate that the Session ID of received QUIC packets matches that advertised in the session parameters (Section 10.2.4) before any HTTP-level processing is done. In the case of validation failure, the receiver SHOULD ignore the packet in order to protect itself from denial-of-service attacks.

2.4. Session Security

The QUIC cryptographic handshake ([RFC9000] and [RFC9001]) sets out methods to achieve the goals of authenticated key exchange and QUIC packet protection between two endpoints forming a QUIC connection. The design facilitates low-latency connection; 1-RTT or 0-RTT. This specification replaces the in-band security handshake, achieving similar goals through the use of session parameters described in Section 3.1.

Integrity and authenticity concerns are addressed in Section 6.1 and Section 6.2 respectively. In order to protect themselves from attack vectors, endpoints SHOULD NOT participate in sessions for which they cannot establish reasonable confidence over the cipher suite or key in use for that session. Participants MAY leave any session that fails to successfully match anticipated security characteristics.

3. Session Advertisement

In this specification, connection negotiation is replaced with a session advertisement mechanism based on HTTP Alternative Services (Alt-Svc) [RFC7838]. This document specifies how the parameters of a multicast QUIC session are expressed as Alt-Svc parameters. The following sections provide a high-level view of these; further details are provided in Section 10.2, with examples provided in Appendix B.1. QUIC connection parameters not defined as, or related to, Alt-Svc parameters are not used.

The definition of a session (including the session ID and its parameters) is not the responsibility of any endpoint. Rather, endpoints SHOULD use session advertisements to determine if they are capable of participating in a given session. This document does not specify which party is responsible for defining and/or advertising multicast QUIC sessions.

Endpoints SHOULD NOT become participants in sessions where the advertisement requirements set out in the present document are unfulfilled.

The freshness of Alt-Svc multicast QUIC session advertisements is as described in section 2.2 of [RFC7838].

It is RECOMMENDED that session advertisements are carried over a secure transport (such as HTTPS) which can guarantee the authenticity and integrity of the Alt-Svc information. This addresses some of the concerns around the protection of session establishment described in Section 11.2.

> **Authors' Note:** We invite review comments on mandating the use of a secure transport for advertising sessions.

Senders MAY also advertise the availability of alternative sessions by carrying Alt-Svc in a multicast QUIC session.

3.1. Security Context

This specification replaces the in-band security handshake. The session parameters "cipher suite", "key" and "iv" (described below) allow for the establishment of a security context. In order to protect themselves, endpoints SHOULD NOT participate in sessions for which they cannot establish reasonable confidence over the cipher suite, key, or IV in use for that session. Endpoints SHOULD leave any sessions which fail to successfully match anticipated security characteristics.

3.1.1. Cipher Suite

Cipher suite negotiation is replaced with a "cipher suite" session parameter, which is advertised as the Alt-Svc parameter cipher-suite (Section 10.2.1).

The Alt-Svc "cipher-suite" parameter is OPTIONAL. If present, this parameter MUST contain only one value that corresponds to an entry in the TLS Cipher Suite Registry (see http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4). Session advertisements that omit this parameter imply that the session is operating with cipher suite 0x00,0x00 (NULL_WITH_NULL_NULL).

3.1.2. Key Exchange

Key exchange is replaced with a "key" session parameter, which is advertised as the Alt-Svc parameter key (Section 10.2.2). The parameter carries a variable-length hex-encoded key for use with the session cipher suite.

The Alt-Svc "key" parameter is OPTIONAL. Session advertisements that omit this parameter imply that the key may be available via an out-of-band method not described in this document.

3.1.3. Initialization Vector

Initialization Vector (IV) exchange is replaced with an "iv" session parameter, which is advertised as the Alt-Svc parameter iv (Section 10.2.3). The parameter carries a variable-length hex-encoded IV for use with the session cipher suite and key.

The Alt-Svc "iv" parameter is OPTIONAL. Session advertisements that omit this parameter imply that the IV may be available via an out-of-band method not described in this document.

3.2. Session Identification

[RFC9000] specifies how the QUIC connection identifiers are used, in particular the independent selection of these identifiers by each endpoint for its peer. In a unidirectional multicast environment, there is no meaningful way for an endpoint to generate a connection identifier for its peer to use. This document defines a "session identifier" session parameter, which is advertised as the Alt-Svc parameter "session-id" (Section 10.2.4). The requirements for the usage of session identifiers have already been described in Section 2.3.

The Alt-Svc "session-id" parameter is optional. Session advertisements MAY contain at most one instance of a "session-id" parameter. Session advertisements that identify the same Any Source Multicast group {G} or Source Specific Multicast group {S,G} indicate that multiple sessions are multiplexed in the same multicast group and each such advertisement must carry a unique "session-id".

3.3. Session Idle Timeout

Conventional QUIC connections may be implicitly terminated following a period of idleness (lack of network activity). The optional QUIC TransportParameter max_idle_timeout provides a means for endpoints to specify the timeout period. This document defines a "session idle timeout" session parameter, which is advertised as the Alt-Svc parameter "session-idle-timeout" (Section 10.2.5). This session parameter mimics the behaviour of max_idle_timeout, providing a means for multicast QUIC sessions to define their own idle timeout periods.

Session idle timeout may be prevented by keep-alive strategies Section 4.10.

The Alt-Svc "session-idle-timeout" parameter is optional. Session advertisements MAY contain zero or more instances of this parameter. If it is repeated, the first occurrence MUST be used and subsequent occurrences MUST be ignored. Session advertisements that omit the "session-idle-timeout" parameter, or set it to zero never time out.

Receiving participants SHOULD leave multicast QUIC sessions when the session idle timeout period has elapsed (Section 4.7). Leaving participants MUST use the silent close method, in which no QUIC CONNECTION_CLOSE frame is sent.

3.4. Session Peak Flow Rate

[RFC9000] specifies a credit-based stream- and connection-level flow control scheme which prevents a fast sender from overwhelming a slow receiver at the stream level, as well as an aggregate level of all streams. Window size connection parameters are exchanged on

connection establishment using the required QUIC TransportParameters initial_max_data, initial_max_stream_data_bidi_local, initial_max_stream_data_bidi_remote and initial_max_stream_data_uni. In a unidirectional multicast environment, such a scheme is infeasible.

This document defines a "peak flow rate" session parameter, expressed in units of bits per second, which is advertised as the Alt-Svc parameter "peak-flow-rate" (Section 10.2.7). This completely replaces the transport parameters listed above, instead indicating the maximum bit rate of QUIC payloads transmitted on all multicast groups comprising the session. It applies at the aggregate level, and is not specific to any single stream.

The Alt-Svc "peak-flow-rate" parameter is OPTIONAL. If the parameter is repeated the first occurrence MUST be used and subsequent occurrences MUST be ignored. Session advertisements that omit the parameter imply that the flow rate is unlimited.

A multicast sender SHOULD NOT cause the advertised peak flow rate of a session to be exceeded. A receiver MAY leave any session where the advertised peak flow rate is exceeded.

3.5. Resource Concurrency

[RFC9000] considers concurrency in terms of the number of active incoming streams, which is varied by the receiving endpoint adjusting the maximum Stream ID. The initial value of maximum Stream ID is controlled by the relevant required QUIC TransportParameters initial_max_streams_bidi and initial_max_streams_uni. They are increased during the lifetime of a QUIC connection by the QUIC MAX_STREAMS frame. In a unidirectional multicast environment, there is no way for a receiver to specify an initial limit nor to increase it. Therefore in multicast QUIC, the maximum Stream ID (initial and always) is 2^62. This mechanism is not used to manage concurrency in multicast QUIC.

Due to the profiling of maximum Stream ID, there is no role for the QUIC STREAMS_BLOCKED frame and it is prohibited. Participants MUST NOT send this frame type. Reception of this frame type MUST be handled as described in Section 4.12.

This document specifies a "maximum concurrent resources" session parameter, which is advertised as the Alt-Svc parameter "max-concurrent-resources" (Section 10.2.6). This parameter replaces initial_max_stream_id_bidi and initial_max_stream_id_uni. It advertises the maximum number of concurrent active resources generated by a sender in a given multicast QUIC session.

The Alt-Svc "max-concurrent-resources" parameter is OPTIONAL. If the parameter is repeated the first occurrence MUST be used and subsequent occurrences MUST be ignored. Session advertisements that omit the parameter imply that the maximum concurrency is unlimited.

A multicast sender participating in a session MUST NOT cause the advertised "max-concurrent-resources" to be exceeded. A receiver MAY leave any session where the advertised limit is exceeded, in order to protect itself from denial-of-service attacks.

3.6. Additional TransportParameter Considerations

> **Authors' Note:** This section will consider TransportParameters that have not already been addressed, as required.

Section 19.21 of [RFC9000] defines a mechanism for endpoints to show willingness to receive one or more extension frame types. It is not possible for multicast QUIC receivers to signal this information to senders.

This document defines an "extensions" session parameter, which is advertised as the Alt-Svc parameter "extensions" Section 10.2.10 and replaces the transport parameter exchange detailed above. The Alt-Svc "extensions" parameter is optional. Session advertisements MAY contain zero or more instances of this parameter. The parameter lists transport parameter values present in the QUIC Transport Parameter Registry as specified in Section 22.3 of [RFC9000].

Only transport parameters which expressly reference Multicast QUIC are considered valid extension parameters.

> **Authors' Note:** The authors welcome suggestions for how to map these extension types more cleanly into this document.

Participants SHOULD NOT join sessions advertising extensions that they do not support, as QUIC frames are not self-describing.

3.7. Digest Algorithm

A method to provide content integrity is described in Section 6.1. This specifies the means to convey a value computed by a particular digest algorithm. The identity of the selected algorithm is also indicated. Valid digest algorithms are collected in the IANA HTTP Digest Algorithm Values registry (http://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml#http-dig-alg-1).

This document specifies a "digest algorithm" session parameter, which is advertised as the Alt-Svc parameter "digest-algorithm" (Section 10.2.8).

> **Authors' Note:** Section 6.1 contains an author's note on the potential for content integrity to become mandatory. This section will be updated in line with the outcome of that decision.

The Alt-Svc "digest-algorithm" parameter is OPTIONAL. Repetition of the "digest algorithm" parameter in a single advertisement describes an algorithm set that MAY be used across the session. Session advertisements that omit the Alt-Svc parameter digest-algorithm imply that either:

* the session does not use the content integrity mechanism, or

* the algorithm set is unrestricted, i.e. a sender may vary the algorithm as it so chooses. This may lead to undesirable results if receivers do not support a chosen algorithm.

Advertising the algorithm set for a session gives receivers the opportunity to selectively join sessions where the algorithms are known to be supported. This may help to mitigate latency issues in the receiver resulting from joining a session only to discover some of its parameters are not supported.

A multicast sender participating in a session MUST NOT use algorithms outside the signalled digest algorithm set. A receiver MAY leave any session where an algorithm outside the digest algorithm set is used.

3.8. Signature Algorithm

A method to provide content authenticity is described in Section 6.2. This specifies the means to convey a value computed by a particular signature algorithm. The identity of the selected algorithm is also indicated. Valid signature algorithms are collected in the IANA Signature Algorithms registry (http://www.iana.org/assignments/signature-algorithms).

This document specifies a "signature algorithm" session parameter, which is advertised as the Alt-Svc parameter signature-algorithm (Section 10.2.9).

> **Authors' Note:** Section 6.2 contains an author's note on the potential for content authenticity to become mandatory. This section will be updated in line with the outcome of that decision.

The Alt-Svc "signature-algorithm" parameter is OPTIONAL. Repetition of the "signature algorithm" parameter in a single advertisement describes an algorithm set that MAY be used across the session.

Session advertisements that omit the Alt-Svc parameter signature-algorithm imply that either:

- the session does not use the content authenticity mechanism, or

- the algorithm set is unrestricted i.e. a sender may vary the algorithm as it so chooses. This may lead to undesirable results if receivers do not support a chosen algorithm.

Advertising the algorithm set for a session gives receivers the opportunity to selectively join sessions where the algorithms are known to be supported. This may help to mitigate latency issues in the receiver resulting from joining a session only to discover some of its parameters are not supported.

A multicast sender participating in a session MUST NOT use algorithms outside the signalled signature algorithm set. A receiver MAY leave any session where an algorithm outside the signature algorithm set is used.

4. QUIC Profile

The profile of [RFC9000] is presented in this section. In order to preserve compatibility with conventional QUIC, the specification works with a limited scope of change. However, the nature of unidirectional multicast communications means that some protocol procedures or behaviours need to be modified.

Section 5.3 of [RFC9000] defines a set of required actions that a QUIC server and QUIC client must be able to perform. Due to the limitations of this profile, all of the requirements in Section 5.3 of [RFC9000] are removed except for:

- Configuring the minimum and total number of permitted streams of each type is described in Section 3.5.

- Multicast QUIC senders may still send PING frames to stop a session from expiring as described in Section 4.10.

4.1. Packet Size

The means for determining an appropriate size for QUIC packets are described in Section 14 of [RFC9000]. Implementations of this specification SHOULD bear in mind that the Path Maximum Transmission Unit (PMTU) may be affected by multicast IP technologies such as Automatic Multicast Tunneling (AMT) [RFC7450]. Additionally, consideration should be given toward the applicability of maximum transmission unit discovery methods (such as PLPMTUD [RFC4821] and PMTUD [RFC1191]) to multicast IP.

4.2. Packet Format

Endpoints implementing this specification MUST only send QUIC packets with the short header form. As short header packets do not include a length, senders MUST NOT attempt to coalesce any QUIC packets in the same UDP datagram. Therefore, all UDP datagrams sent by senders conforming to this profile contain exactly one QUIC packet.

4.2.1. Packet Numbers

All packets for this profile SHALL be numbered in the application data packet number space. The initial and handshake packet number spaces are not used by this profile, as the handshake is replaced by an out-of-band mechanism (see Section 2.4).

The encoding of packet numbers in QUIC packets is described in Section 17.1 of [RFC9000]. Senders must always use the same number of bytes to represent the packet number for all packets sent to a session. Because a receiver may join a session after the sender has already sent several packets, it MUST NOT assume that the first packet number will be 0.

4.2.2. Spin Bit

[RFC9000] specifies a bit in the 1-RTT packet header as the latency spin bit that may be used to measure network round trip latency between a client and a server. This mechanism is not usable in a unidirectional multicast packet flow. Senders SHALL set the spin bit to zero in all packets. Receivers SHOULD ignore the spin bit.

> **Authors' Note:** The authors welcome suggestions for the use of the spin bit in a multicast context.

4.3. Connection Identifier

The Destination Connection ID field MUST be non-zero-length in every QUIC packet if the session was advertised with a session-id session parameter (Section 10.2.4). If the multicast QUIC session advertisement does not carry a "session identifier" session parameter, then the Destination Connection ID MUST be zero-length in any QUIC packet for that session. In the case where multiple sessions are multiplexed on the same 5-tuple network association, the Destination Connection ID field MUST be non-zero-length in every QUIC packet and must be distinct for each session.

4.4. Stream Identifier

The maximum Stream ID of a multicast QUIC session is 2^62, as explained in Section 3.5. With the exception of the first client-initiated request Stream ID, which is reserved as described in

Section 5.2, all Stream ID values SHALL be of the server-initiated unidirectional stream type.

4.5. Flow Control

Conventional QUIC provides stream- and connection-level flow control, and endpoints manage this by sending QUIC MAX_DATA or MAX_STREAM_DATA frames as required. When a sender is blocked from sending flow-controlled frames, it sends an informational QUIC DATA_BLOCKED or STREAM_DATA_BLOCKED frame.

In a unidirectional environment, the sender never has a receive window and the receiver cannot send in-band updates. Therefore, the management of flow-control windows and transmission of blockage information is not supported by this profile. The QUIC MAX_DATA, MAX_STREAM_DATA, DATA_BLOCKED and STREAM_DATA_BLOCKED frames are prohibited by this profile. Participants MUST NOT send these frame types. Reception of these frame types MUST be handled as described in Section 4.12.

4.6. Stream Termination

A sender MAY prematurely terminate the transmission on any unreserved QUIC Stream ID by setting the FIN bit of a QUIC STREAM frame, or by sending a QUIC RESET_STREAM frame (as specified in [RFC9000] and [RFC9114]).

Receiving participants MUST NOT make any attempt to send QUIC RESET_STREAM frames to the multicast group.

4.7. Connection Shutdown

Explicit shutdown of a multicast QUIC session using QUIC methods is not supported by this profile.

The QUIC CONNECTION_CLOSE frames and the Stateless Reset packet are prohibited. Participants MUST NOT send these and reception MUST be handled as described in Section 4.12.

Explicit session tear-down using HTTP semantics is allowed, as described in Section 5.4.

Implicit shutdown by means of silent close is also supported, as described in Section 3.3.

4.8. Connection Migration

[RFC9000] has a connection migration feature that allows a connection to survive changes to endpoint addresses. This profile does not currently support connection migration, and as such the QUIC

NEW_CONNECTION_ID and RETIRE_CONNECTION_ID frames are prohibited. Similarly, the QUIC PATH_CHALLENGE and PATH_RESPONSE frames are also prohibited, but additionally because they require bidirectional capability that this profile does not provide.

Endpoints participating in a session conforming to this profile MUST only use a single session ID for the duration of the session, and as such there is no mapping for the active_connection_id_limit transport parameter specified in section 5.1.1 of [RFC9000] in this profile.

> **Author's Note**: Seamless migration from one multicast QUIC session to another is described in Section 2.1.3.

4.9. Explicit Congestion Notification

[RFC9000] specifies that clients may use Explicit Congestion Notification (ECN) [RFC3168]. ECN allows receivers to inform senders of impending congestion before packets are dropped, and the sender may then reduce its transmission rate. As ECN requires bidirectional communication for the receiver to inform the sender of the congestion, the use of ECN for this profile is prohibited.

> **Author's Note**: It is the responsibility of a receiver to determine whether network conditions permit the uncongested reception of a given session based on the advertised peak-flow-rate parameter.

4.10. Session Keep-alive

The flow of traffic in a multicast QUIC session is driven by a sender. There may be periods where the sender has no application data to send for a period longer than the session idle timeout. This profile repurposes the QUIC PING frame to act as a unidirectional keep-alive message that may be sent in order to inform receivers that the session should remain in the Fully-established state. [RFC9000] contains guidance on the sending frequency of QUIC PING frames.

Senders MAY send a QUIC PING frame at any time in order to inform receivers that the session traffic flow has not fallen idle. This frame MUST NOT be acknowledged. Indeed, QUIC ACK frames are prohibited by this profile (Section 4.11).

Receiving participants MUST NOT make any attempt to send QUIC PING frames.

4.11. Loss Detection and Recovery

Receivers implementing this profile MUST NOT make any attempt to acknowledge the reception of QUIC packets. The QUIC ACK frame is prohibited for both senders and receivers. Reception of this frame MUST be handled as described in Section 4.12.

Section 7 specifies alternative strategies for loss recovery.

4.12. Prohibited QUIC Frames and Packets

The following QUIC packets MUST NOT be transmitted by participants: Any packets with a long header (Initial, 0-RTT Protected, Handshake, Retry), Version Negotiation, Stateless Reset.

The following QUIC frames MUST NOT be transmitted by participants: ACK, CONNECTION_CLOSE, CRYPTO, DATA_BLOCKED, HANDSHAKE_DONE, MAX_DATA, MAX_STREAM_DATA, MAX_STREAMS, NEW_CONNECTION_ID, NEW_TOKEN, PATH_CHALLENGE, PATH_RESPONSE, RETIRE_CONNECTION_ID, STOP_SENDING, STREAM_DATA_BLOCKED, STREAMS_BLOCKED.

In addition, any QUIC extension frames not advertised in the session advertisement Section 3.6 MUST NOT be transmitted by participants.

The following QUIC frames MUST NOT be transmitted by receivers: PING, RESET_STREAM.

Reception of a prohibited or non-advertised QUIC frame or packet is a protocol error. Receivers MUST ignore all prohibited QUIC frames and packets.

5. HTTP/3 Profile

HTTP over multicast QUIC depends on HTTP server push, as described in Section 4.6 of [RFC9114]. Section 5.2 below applies an additional constraint on the use of server push. A multicast sender participating in a session pushes resources as a series of QUIC STREAM frames carrying HTTP/3 PUSH_PROMISE, HEADERS and DATA frames. Examples of this are provided in Appendix B.2. Senders MUST comply with the requirements of the session parameters, as described earlier in Section 3.

The profile of HTTP/3 specified in this section places additional constraints on the use of metadata compression (Section 5.3).

5.1. HTTP Connection Settings

The HTTP/3 SETTINGS frame is prohibited by this profile. Participants MUST NOT make any attempt to send this frame type. Reception of this frame MUST be handled as described in Section 5.7.

5.2. Server Push

Server push is, by default, disabled for HTTP/3 connections. A conventional HTTP/3 client enables and manages server push by controlling the maximum Push ID ([RFC9114], Section 7.2.7), achieved by sending the HTTP/3 MAX_PUSH_ID frame.

This profile mandates the use of server push, and specifies no means to disable it. The maximum Push ID for multicast QUIC sessions (initial and always) is 2^62. Values of Push ID SHALL be allocated in accordance with [RFC9114].

Server push concurrency in multicast QUIC is described in Section 3.5. There is no role for the HTTP/3 MAX_PUSH_ID frame and it is prohibited. Participants MUST NOT send this frame type. Reception of this frame type MUST be handled as described in Section 5.7.

For this profile, the Stream Type for any new server-initiated unidirectional stream MUST be Server Push (0x01).

The HTTP/3 CANCEL_PUSH frame MAY be used by sending participants to abort sending a response for the identified server push. Usage of this frame SHALL follow the guidance for servers in [RFC9114].

Receiving participants MUST NOT make any attempt to send HTTP/3 CANCEL_PUSH frames to the multicast group.

Receiving participants SHOULD ignore all frames on server-initiated unidirectional streams for which the packet containing the Push ID has not been received. Receiving participants SHOULD ignore all frames on server-initiated unidirectional streams carrying a Push ID that was not previously promised to the receiver.

Conventionally, pushed responses are associated with an explicit request from a client. This is not possible when using a unidirectional transport such as multicast IP. This profile reserves the first client-initiated, bidirectional QUIC stream. HTTP/3 PUSH_PROMISE frames MUST be sent on this reserved Stream ID.

5.3. Metadata Compression

The compression of HTTP header fields is considered in QPACK [RFC9204], which describes two methods for the compression of HTTP header fields: indexing (via static or dynamic tables) and Huffman encoding.

A multicast QUIC session, as described in the present document, does not provide the assurances (receiver participation, transport reliability) required to sufficiently maintain the dynamic decoding context. Therefore, this document requires that endpoints SHALL NOT use dynamic table indexing. It is RECOMMENDED that endpoints use static table indexing and/or Huffman encoding in order to benefit from the remaining compression methods available.

5.4. Session Tear-down

A multicast QUIC session MAY be explicitly torn down by means of the Connection: close HTTP header described in section 6.6 of [RFC7230]. A sender intending to leave the session SHOULD include the Connection: close header in its response metadata. A sender SHOULD transmit all outstanding frames related to remaining request/response exchanges before ending transmission to the multicast group. A receiver SHOULD continue to receive and process frames until all outstanding request/response exchanges are complete.

The HTTP/3 GOAWAY frame is prohibited. Participants MUST NOT send this and reception MUST be handled as described in Section 5.7.

5.5. Effects of Packet Loss

Since multicast QUIC is an inherently unreliable delivery mechanism, portions of HTTP messages sent by the sender may not be received by the receiver. Conventional HTTP/3 frames assume reliable, in-order delivery by the underlying QUIC stream. HTTP/3 frames do not therefore carry any information indicating their position within the stream, because this information is instead carried in the QUIC STREAM frame header.

In multicast QUIC, packet loss leaves gaps in the flow of a stream, and therefore gaps in the HTTP/3 frame(s) that are carried on that stream.

1. Loss of an HTTP/3 PUSH_PROMISE frame renders the reception of an entire stream impossible, since the receiver ignores the new push stream that has not been promised as described in Section 5.2.

2. Loss of a QUIC packet comprising all or part of an HTTP/3 HEADERS frame will likely prevent the QPACK decoder from successfully parsing the received metadata, leaving the receiver unable to make use of the affected HTTP representation.

3. Loss of a QUIC packet containing an HTTP/3 DATA frame header leaves a receiver unsure where to look for the start of the next HTTP/3 frame on the same stream. In the unlikely event that the receiver manages to resynchronise to the start of a subsequent HTTP/3 DATA frame, the position of the payload in the HTTP representation data is unknown because it is not explicitly signalled in the DATA frame header. The Offset field in the QUIC STREAM header describes the offset within the stream, but it is impossible for the receiver to know the size of any DATA frame headers that were lost.

4. Loss of QUIC packets that only contain part of an HTTP/3 DATA frame Data field (i.e. beyond the frame header) can be recovered using the unicast repair mechanism described in Section 7.2.

[H3-DATA-OFFSET] describes the DATA_WITH_OFFSET HTTP/3 extension frame type, which can be used to solve the third problem described above. Multicast QUIC sessions that make use of the DATA_WITH_OFFSET extension frame MUST advertise the session with the appropriate non-compatible experiment identifier "data-offset" in the ALPN string, as specified in Section 9.1.

5.6. HTTP/3 Extension frames

Except where explicitly allowed by this specification (e.g. in Section 5.5 above), HTTP/3 extension frames (e.g. ALTSVC) are prohibited by this profile. Participants MUST NOT make any attempt to send prohibited extension frame types. Reception of these MUST be handled as described in Section 5.7.

5.7. Prohibited HTTP/3 Frames

The following HTTP/3 frames MUST NOT be transmitted by participants: GOAWAY, MAX_PUSH_ID, SETTINGS.

In addition, all HTTP/3 extension frame types MUST NOT be transmitted by participants.

The following HTTP/3 frames MUST NOT be transmitted by receivers: CANCEL_PUSH.

Reception of a prohibited HTTP/3 frame is a protocol error. Receivers MUST ignore prohibited HTTP/3 frames.

6. Application-Layer Security

As already described in Section 3.1, the implicit cipher suite used by a multicast QUIC session makes very limited provision for security in the transport and session layers. This section profiles the use of some additional features to provide equivalent functionality at the application-layer.

6.1. Content Integrity

In many applications, it is important to ensure that an HTTP representation has been received intact (i.e. has not suffered from transmission loss or random bit errors) before passing the received object on to the receiving application. A mechanism is therefore specified here to provide end-to-end content integrity protection for

HTTP representations in transit. The use of this content integrity mechanism is RECOMMENDED.

> **Authors' Note:** We invite review comments on mandating the use of this content integrity mechanism.

[RFC3230] specifies an instance digest HTTP header called Digest. A sender MAY include this header in the HTTP/3 HEADERS frame of any representation it transmits and a receiver MAY use this header to validate the integrity of the received representation once it has been reassembled. Where this validation fails, the receiver SHOULD discard the representation without processing it further.

Note that the digest value protects a whole HTTP instance (i.e. the representation of a resource at the point of transmission as opposed to the body of a particular HTTP message). In cases where partial representations are fragmented over one or more HTTP response messages, the digest value is computed over the complete representation prior to fragmentation into partial responses.

Any of the algorithms specified in the IANA registry of digest algorithms (http://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml#http-dig-alg-1) MAY be used in conjunction with the Digest header. There is no requirement for participants to support the full set of algorithms.

6.2. Content Authenticity

In some applications, it is important for a receiver to reassure itself that an HTTP representation has been received from an authentic source. It is also sometimes useful for a receiver to know that the information has not been tampered with in transit by a malicious intermediate actor. A mechanism is therefore specified here to prove the authenticity of HTTP messages in transit. The use of this content authenticity mechanism is RECOMMENDED for senders implementing this specification.

> **Authors' Note:** We invite review comments on mandating the use of this content authenticity mechanism.

[I-D.cavage-http-signatures] specifies a means of securely signing metadata associated with any HTTP message. The resulting digital signature is conveyed in the Signature header of the message itself. The Signature header also conveys a list of HTTP header fields over which the signature was computed. A receiver MAY verify the Signature header in order to validate the authenticity of received metadata. Where this validation fails, the receiver SHOULD discard or ignore any related metadata and/or data without processing it further.

Note that the signature proves the authenticity of the metadata in a single HTTP message. A Signature header MAY be included separately in the HTTP/3 PUSH_PROMISE frame (protecting the request metadata) and in the final (or only) HTTP/3 HEADERS frame relating to a particular resource (protecting the response metadata). In order to provide an additional level of protection, however, it is RECOMMENDED that the signature be computed over the combined request metadata (from the HTTP/3 PUSH_PROMISE frame) and the corresponding response metadata (from the HTTP/3 HEADERS frames) of the same resource. This binds the request metadata and response metadata together, providing the receiver with additional reassurance of authenticity. In this latter case, the combined digital signature SHALL be conveyed in the final (or only) HTTP/3 HEADERS frame.

In applications where the detection of replay attacks is a requirement, it is RECOMMENDED that the Date header be included in the scope of the signature. It is RECOMMENDED that receivers use the value of the Date header for replay detection using appropriate strategies (e.g. checking for freshness). The definition of such strategies is beyond the scope of this document.

In applications where the authenticity and integrity of the transmission are both important, it is RECOMMENDED that the Digest header specified in Section 6.1 above is included in the scope of the signature. By signing the instance digest, the authenticity and integrity of the HTTP message body are also assured in addition to that of the metadata.

Any of the algorithms specified in the IANA registry of signature algorithms (http://www.iana.org/assignments/signature-algorithms) MAY be used in conjunction with the Signature header. There is no requirement for participants to support the full set of algorithms.

6.3. Content Confidentiality

In applications where there is a requirement for the content itself to remain confidential, appropriate steps SHOULD be taken to protect the application payload, for example by encrypting it. This document does not preclude the use of application-level encryption, but does not specify a mechanism for the distribution of content decryption keys.

7. Loss Recovery

Because the acknowledgement of received packets to multicast groups is prohibited by this specification (Section 4.11) the detection of discarded or corrupted packets is the sole responsibility of the receiver, and such losses must be recovered by means other than the retransmission mechanism specified in [RFC9000] and [RFC9002].

7.1. Forward Error Correction

> **Authors' Note:** A simple parity-based Forward Error Correction scheme was removed from the experimental QUIC wire protocol specification in version Q032.

A sender MAY make use of a suitable Forward Error Correction scheme to allow a receiver to reconstruct lost packets from those that have been successfully received.

7.2. Unicast Repair

In the case where a lost QUIC packet cannot be recovered using Forward Error Correction, either because the number of packets lost exceeds the scheme's threshold for reconstruction, or because FEC is not in use on the multicast QUIC session, a receiver MAY instead recover the missing payload data using conventional unicast HTTP requests to an origin server.

- The total size of the resource is indicated in the content-length response header carried in the HTTP/3 HEADERS frame.

- The location of the missing data can be determined by examining the Offset field in the QUIC STREAM frame headers of successfully received QUIC packets.

Using this information, a receiver MAY compose an efficient HTTP range request [RFC7233] to the origin server indicated in the URL. Several disjoint ranges MAY be combined into a single HTTP request. A receiver MAY direct its request to an alternative server using Alt-Svc information received on the multicast QUIC session, or else received as part of a previous unicast HTTP response according to the rules in [RFC7838].

8. Transmission of Partial Content

Under certain circumstances, a sender may not be in full possession of a resource body when transmission begins, or may not be able to guarantee that a transmission will complete. In such cases, the sender MAY employ the syntax of an HTTP range request [RFC7233] to indicate partial content, as specified below. All receivers SHALL implement support for such HTTP range requests.

If partial content is to be transmitted:

- The range header (Section 3.1 of [RFC7233]) SHALL be present in the HTTP/3 PUSH_PROMISE frame.

*The corresponding HTTP/3 HEADERS frame SHALL indicate HTTP status code 206.

-The range being transmitted SHALL be indicated in a content-range header field and the size of the complete resource indicated in a content-length header field.

9. Protocol Identifier

The HTTP over multicast QUIC protocol specified in this document is identified by the application-layer protocol negotiation (ALPN) [RFC7301] identifier "h3m". The IANA registration of this protocol identifier can be found in Section 12.1. This reserves the ALPN identifier space but describes a protocol that does not use TLS. The usage of the "h3m" identifier for discoverability is described in Section 10.

9.1. Draft Version Identification

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Only implementations of the final, published RFC can identify themselves as "h3m". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier. For example, draft-pardue-quic-http-mcast-06 is identified using the string "h3m-06".

Non-compatible experiments that are based on these draft versions MUST append the string "-" and an experiment name to the identifier. For example, an experimental implementation based on draft-pardue-quic-http-mcast-06 which uses extension features not registered with the appropriate IANA registry might identify itself as "h3m-06-extension-foo". Note that any label MUST conform to the "token" syntax defined in Section 3.2.6 of [RFC7230]. Experimenters are encouraged to coordinate their experiments.

10. Discovery of Multicast QUIC Sessions

The announcement and discovery of services operating over multicast IP has previously been specified by the Session Description Protocol (SDP) [RFC4566], Session Announcement Protocol [RFC2974] and Session Initiation Protocol [RFC3261]. These are typically deployed together and in conjunction with a multicast-friendly transport such as the Real-time Transport Protocol (RTP) [RFC3550].

In contrast, the present document specifies a mechanism for advertising services that is built into HTTP metadata and is consistent across unicast and multicast resource delivery modes. This means that a single application-layer can be used for service advertisement/discovery, and for bulk data transmission/reception. Specifically, the Alt-Svc HTTP header is specified as the means to advertise multicast services from a unicast service. A unicast HTTP response MAY be decorated with an Alt-Svc value that hints to the client about the availability of the resource via a multicast QUIC session. A client that supports such a multicast QUIC session MAY then transparently switch to it.

Symmetrically, the Alt-Svc header can also be used to advertise the unicast service from a multicast service. A resource transmitted as part of a multicast QUIC session MAY be decorated with an Alt-Svc value that hints to the client about the availability of the resource via an alternative unicast HTTP server. A receiver MAY then use this HTTP server for unicast resource patching (Section 7.2).

Where HTTP over multicast QUIC sessions are advertised using Alt-Svc, the protocol identifier SHALL be "h3m", as specified in Section 9.

10.1. Source-specific Multicast Advertisement

Source-specific multicast (SSM) [RFC4607] MAY be used for the delivery of multicast services.

> **Authors' Note:** We invite review comments on mandating the use of source-specific multicast only.

This document specifies the "source-address" parameter for Alt-Svc, which is used to provide the SSM source address to endpoints.

Syntax:

```
source-address = uri-host; see RFC7230, Section 2.7
```

For example:

```
source-address=192.0.2.1
```

When a multicast QUIC session is provided using SSM, the source-address parameter MUST be advertised. If the parameter is repeated, the first occurrence MUST be used and subsequent occurrences MUST be ignored.

10.2. Session Parameter Advertisement

The concept of session parameters is introduced in Section 2.2. This section details how the session parameters are expressed as Alt-Svc parameters.

10.2.1. Cipher Suite

This document specifies the "cipher-suite" parameter for Alt-Svc, which carries the cipher suite in use by a multicast QUIC session. cipher-suite MUST contain one of the values contained in the TLS Cipher Suite Registry (http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4):

Syntax:

```
cipher-suite = 4*4 HEXDIG
```

For example, the following specifies cipher suite 0x13,0x01 (TLS_AES_128_GCM_SHA256):

```
cipher-suite=1301
```

The requirements for endpoint usage of cipher-suite are described in Section 3.1.

10.2.2. Session Key

This document specifies the "key" parameter for Alt-Svc, which carries the cryptographic key in use by the multicast QUIC session.

Syntax:

```
key = *HEXDIG
```

For example:

```
key=4adf1eab9c2a37fd
```

The requirements for endpoint usage of key are described in Section 3.1.

10.2.3. Session Cipher Initialization Vector

This document specifies the "iv" parameter for Alt-Svc, which carries the cipher Initialization Vector (IV) in use by the multicast QUIC session.

Syntax:

```
iv = *HEXDIG
```

For example:

```
iv=4dbe593acb4d1577ad6ba7dc3189834e
```

The requirements for endpoint usage of iv are described in Section 3.1.

10.2.4. Session Identification

This document defines the "session-id" parameter for Alt-Svc, which carries the multicast QUIC session identifier.

Syntax:

```
session-id = *HEXDIG
```

For example, the following specifies session 101 (0x65 hexadecimal):

```
session-id=65
```

The requirements for endpoint usage of session-id are described in Section 2.3. In the above example, the Destination Connection ID field in every QUIC packet header would be one byte in size. For a session-id of BADBEEF then then Destintation Connection ID field in every QUIC packet header would be four bytes in size.

10.2.5. Session Idle Timeout Period

This document specifies the "session-idle-timeout" parameter for Alt-Svc, which carries the idle timeout period in milliseconds of a multicast QUIC session.

Syntax:

```
session-idle-timeout = *DIGIT ; integer milliseconds
```

For example, the following specifies a one-minute session idle timeout period:

```
session-idle-timeout=60
```

The requirements for endpoint usage of session-idle-timeout are described in Section 3.3.

10.2.6. Resource Concurrency

This document specifies the "max-concurrent-resources" parameter for Alt-Svc, which expresses the maximum number of concurrent active resources from the sender in a multicast QUIC session.

Syntax:

```
max-concurrent-resources = *DIGIT ; unsigned 32-bit integer
```

For example, the following specifies that no more than 12 (decimal) resources will be concurrently active in the session:

```
max-concurrent-resources=12
```

The requirements for endpoint usage of max-concurrent-resources are described in Section 3.5.

10.2.7. Session Peak Flow Rate

This document specifies the "peak-flow-rate" parameter for Alt-Svc, which expresses the expected maximum aggregate transfer rate of data from all sources of the multicast QUIC session.

Syntax:

```
peak-flow-rate = *DIGIT ; bits per second
```

For example, the following specifies a peak flow rate of 550 kbits/s in the session:

```
peak-flow-rate=550000
```

The requirements for endpoint usage of peak-flow-rate are described in Section 3.4.

10.2.8. Digest Algorithm

This document specifies the "digest-algorithm" parameter for Alt-Svc, which carries the digest algorithm in use by a multicast QUIC session. digest-algorithm MUST contain one of the values defined in the HTTP Digest Algorithm Values registry (https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml#http-dig-alg-1).

Syntax:

```
digest-algorithm = token
```

For example, the following specifies a digest algorithm of SHA-256:

```
digest-algorithm=SHA-256
```

The requirements for endpoint usage of digest-algorithm are described in Section 3.7.

10.2.9. Signature Algorithm

This document specifies the "signature-algorithm" parameter for Alt-Svc, which carries the signature algorithm in use by a multicast QUIC session. signature-algorithm MUST contain one of the values defined in the Signature Algorithms registry (http://www.iana.org/assignments/signature-algorithms).

Syntax:

```
signature-algorithm = token
```

For example, the following specifies a signature algorithm of SHA-256:

```
signature-algorithm=rsa-sha256
```

The requirements for endpoint usage of signature-algorithm are described in Section 3.8.

10.2.10. Extensions

This document specifies the "extensions" parameter for Alt-Svc, which carries a list of extension types potentially in use by a multicast QUIC session. extensions MUST only contain values from the QUIC Transport Parameter registry ([RFC9000], section 22.3) that have explicit support for multicast QUIC. Each entry in the list consists of a key identifying the transport parameter, and an optional value. Both the key and the value are hex-encoded.

Syntax:

```
extensions           = DQUOTE ext-transport-param
                       *[ "," ext-transport-param ] DQUOTE
ext-transport-param  = ext-key [ "=" ext-value ]
ext-key              = 4*4HEXDIG; Transport Parameter key
ext-value            = *HEXDIG; Optional Transport Parameter value
```

For example, the following specifies two extensions:

```
extensions="0094,0d0d=f00"
```

The requirements for endpoint usage of extensions are described in Section 3.6

11. Security Considerations

This document specifies a profile of QUIC and HTTP/3 that changes the security model. In order to address this, application-level security methods are described in Section 6. This document does not preclude the use of secure multicast approaches that may provide additional security assurances required for certain use cases.

The use of side-channel or out-of-band technologies (potentially bidirectional interactions) to support multicast QUIC sessions are considered out of this document's scope. Services using such technologies should apply their security considerations accordingly.

11.1. Pervasive Monitoring

Certain multicast deployment architectures may require the use of a session decryption key shared by all participants. Furthermore, the discovery mechanism described in this document provides a means for a receiver to obtain a session decryption key without joining the session. The act of removing packet protection in order to inspect or modify application contents may, in certain deployments, be trivial. The exploration of restricting key learning or session joining to authorised participants goes beyond the scope of this document.

Because in-band multicast interactions are unidirectional, the impact of Pervasive Monitoring [RFC7258] on in-band traffic flows is inherently reduced. Actors can only inspect or modify sender-initiated traffic. Additional measures for content confidentiality may mitigate the impact further. This is discussed in Section 6.3.

Further Pervasive Monitoring concerns are addressed in the following sections.

11.1.1. Large-scale Data Gathering and Correlation

Multicast QUIC sessions decouple sending and receiving participants. Session participation is subject to operations that allow an endpoint to join or leave a multicast group, typically IGMP [RFC3376] or MLD [RFC3810]. The propagation intent of these messages travelling deeper through a network hierarchy generally leads to the anonymisation of data if implemented as specified. It may be possible to gather user-identifiable messages close to the network edge, for example a router logging such messages. However, this would require wide-ranging access across Internet Service Provider networks. Therefore, while such attacks are feasible, it can be asserted that gathering and correlating user-identifiable traffic is difficult to perform covertly and at scale.

11.1.2. Changing Content

Sessions that use a symmetric key for packet protection are subject to the possibility of a malicious actor modifying traffic at some point in the network between a legitimate sender and one (or more) receivers. Receiver-side validation, as specified in Section 6 of the present document, and also in [RFC9000], allows for the detection of such modification. Two approaches help mitigate the impact of modification; the first is application-level methods that protect data (Section 6.1) and metadata (Section 6.2); the second is reduction of the QUIC packet attack surface by means of removal of many frame types (Section 4.12 and Section 5.7).

11.2. Protection of Discovery Mechanism

Multicast QUIC session advertisements SHOULD be conveyed over a secure transport that guarantees authenticity and integrity in order to mitigate attacks related to a malicious service advertisement, for example a "man in the middle" directing endpoints to a service that may lead to other attacks or exploitations.

> **Authors' Note:** We invite review comments on mandating the use of a secure transport for advertising sessions.

Endpoints that make use of multicast QUIC session advertisements SHOULD have reasonable assurance that the alternative service is under control of, and valid for, the whole origin, as described in Section 2.1 of [RFC7838]. Section 6.2 discusses measures that may be used to fulfil this requirement.

11.3. Spoofing

11.3.1. Sender Spoofing

A malicious actor may be able to stage a spoof attack by sending fake QUIC packets to a multicast QUIC session. This could affect the operation or behaviour of receivers. In a multicast scenario, this form of attack has the potential to scale massively.

The feasibility of spoofing a multicast sender is governed by the characteristics of the multicast deployment and network infrastructure. The use of source-specific multicast [RFC4607] may reduce the feasibility. The use of content authenticity (Section 6.2) may mitigate concerns for the application-level messages. However, there remains the possibility for transport-level messages to be spoofed. Multicast applications should consider further mitigations to address this concern.

11.4. Replay Attacks

Conventional QUIC strategies for protecting against replay attacks apply similarly here.

Certain multicast QUIC sessions may use a shared key for transport-level encryption, which would allow an attacker to record, decrypt, repackage and replay QUIC packets. Section 6.2 discusses how the application-level contents may be protected from replay (by signing the Date HTTP header), which provides some mitigation to the success rate or effects of replay attacks.

11.5. Message Deletion

Since HTTP over multicast QUIC is designed to tolerate unreliable delivery, the impacts of message deletion attacks are presumed to be small. Deletion of packets carrying HTTP headers will cause a receiver to ignore subsequent packets carrying body data. Furthermore, the use of multicast QUIC sessions is opportunistic; disruption in service (for example, deleting packets and causing a receiver to fail in construction of a content object) is mitigated by falling back to a unicast service. Considerations for how this may affect the performance of the unicast service are given in Section 11.6.3.

11.6. Denial of Service

11.6.1. Unprotected Frames and Packets

The profile described in the present document provides the means for a multicast sender to protect QUIC packets with a shared key, which is not a strong protection. The weak protection of QUIC packets could present a denial-of-service risk. To mitigate the impact of handling such QUIC packets, certain frames and packets are prohibited as described in (Section 4.12 and Section 5.7).

The frame types that are allowed by this profile do not present a risk of denial of service. Concerns over authenticity and integrity are addressed by the application-layer protection mechanisms described in Section 6.

11.6.2. Network Performance Degradation

The possibility for malfunctioning or malicious participants to degrade the network is a broad issue and considered out of scope for this document. Guidelines and concerns discussed in UDP Best Practices [RFC8085] and other sources apply equally here. This specification does not preclude the use of network performance degradation mitigation solutions such as network circuit breakers.

11.6.3. Unicast Repair Stampeding Herd

Deployments that support the unicast repair mechanism described in Section 7.2 should be aware that a triggering of this behaviour (either deliberate, planned or unplanned) in a large population of multicast receivers may cause a stampeding herd of client requests to the unicast repair service. Service operators SHOULD mitigate the impact of stampeding herd on their deployment.

11.7. Receiver Resource Usage

The application of receiver-side validation, as defined in the present document and in [RFC9000], adds some protection against allocating resource to the processing of bad data.

11.8. Unicast Repair Information Leakage

The unicast repair mechanism may lead to the leakage of user behaviour data. An attacker could gain insight into any receiver participating in a multicast QUIC session, for example by monitoring the TCP port of the unicast alternative. This alone is no worse than current abilities to monitor unicast interactions, for example observing the SNI field contained in a TLS ClientHello. The complete protection of unicast interactions is beyond the scope of this document. However, knowledge that a user (or group of users) has participated in a session is sensitive and may be obtained by correlation between with observable multicast and unicast traffic.

To give an example, a malicious "man in the middle" could purposely cause all receivers to perform a unicast repair (by disrupting the QUIC traffic flow in some way). The disruption is untargeted and may be simple to orchestrate, but the correlation of user activity data, especially across a distributed repair service (e.g. a CDN), requires resources that may reduce the attractiveness of such an attack.

The ability for an attacker to disrupt multicast QUIC sessions is mitigated by this profile (mainly the prohibition of frames and packets). Application-layer security measures described in Section 6 reduce the feasibility further.

Multicast receivers concerned about this form of leakage can eliminate this risk completely by disabling support for unicast repair, at the potential cost of reduced service quality.

12. IANA Considerations

12.1. Registration of Protocol Identification String

This document creates a new registration for the identification of the HTTP over multicast QUIC protocol in the "Application-Layer

Protocol Negotiation (ALPN) Protocol IDs" registry established by [RFC7301].

The "h3m" string identifies HTTP semantics expressed as HTTP mapped to a QUIC layer and carried over IP multicast:

Protocol: Bulk data transport using HTTP over multicast QUIC

Identification Sequence: 0x68 0x33 0x6D ("h3m")

Specification: This document, Section 9

This entry reserves an identifier that is not allowed to appear in TLS Application-Layer Protocol Negotiation.

12.2. Registration of Alt-Svc parameters

This document creates seven registrations for the identification of parameters for the "Hypertext Transfer Protocol (HTTP) Alt-Svc Parameter Registry" established by [RFC7838] (http://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml#alpn-protocol-ids).

12.2.1. Source Address

Parameter name: source-address

Specification: This document, Section 10.1

12.2.2. Cipher Suite

Parameter name: cipher-suite

Specification: This document, Section 10.2.1

12.2.3. Key

Parameter name: key

Specification: This document, Section 10.2.2

12.2.4. Initialization Vector

Parameter name: iv

Specification: This document, Section 10.2.3

12.2.5. Session Identifier

Parameter name: session-id

Specification:
This document, Section 10.2.4

12.2.6. Session Idle Timeout

Parameter name: session-idle-timeout

Specification: This document, Section 10.2.5

12.2.7. Maximum Concurrent Resources

Parameter name: max-concurrent-resources

Specification: This document, Section 10.2.6

12.2.8. Peak Flow Rate

Parameter name: peak-flow-rate

Specification: This document, Section 10.2.7

12.2.9. Digest Algorithm

Parameter name: digest-algorithm

Specification: This document, Section 10.2.8

12.2.10. Signature Algorithm

Parameter name: signature-algorithm

Specification: This document, Section 10.2.9

12.2.11. Extension

Parameter name: extension

Specification: This document, Section 10.2.10

13. References

13.1. Normative References

[H3-DATA-OFFSET] Hurst, S., "An Offset Extension Frame For HTTP/3 Data", Work in Progress, Internet-Draft, draft-hurst-quic-http-data-offset-frame-02, <https://datatracker.ietf.org/doc/html/draft-hurst-quic-http-data-offset-frame-02>.

[I-D.cavage-http-signatures] Cavage, M. and M. Sporny, "Signing HTTP Messages", Work in Progress, Internet-Draft, draft-cavage-http-signatures-12, 21 October 2019, <https://

www.ietf.org/archive/id/draft-cavage-http-signatures-12.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3230] Mogul, J. and A. Van Hoff, "Instance Digests in HTTP", RFC 3230, DOI 10.17487/RFC3230, January 2002, <https://www.rfc-editor.org/info/rfc3230>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/info/rfc4607>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/info/rfc7233>.

[RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <https://www.rfc-editor.org/info/rfc7234>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <https://www.rfc-editor.org/info/rfc7405>.

[RFC7838] Nottingham, M., McManus, P., and J. Reschke, "HTTP Alternative Services", RFC 7838, DOI 10.17487/RFC7838, April 2016, <https://www.rfc-editor.org/info/rfc7838>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9204] Krasic, C., Bishop, M., and A. Frindell, Ed., "QPACK: Field Compression for HTTP/3", RFC 9204, DOI 10.17487/RFC9204, June 2022, <https://www.rfc-editor.org/info/rfc9204>.

13.2. Informative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, DOI 10.17487/RFC2974, October 2000, <https://www.rfc-editor.org/info/rfc2974>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time

Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, DOI 10.17487/RFC5737, January 2010, <https://www.rfc-editor.org/info/rfc5737>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <https://www.rfc-editor.org/info/rfc5740>.

[RFC6676] Venaas, S., Parekh, R., Van de Velde, G., Chown, T., and M. Eubanks, "Multicast Addresses for Documentation", RFC 6676, DOI 10.17487/RFC6676, August 2012, <https://www.rfc-editor.org/info/rfc6676>.

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <https://www.rfc-editor.org/info/rfc6726>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <https://www.rfc-editor.org/info/rfc7450>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/

RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8866] Begen, A., Kyzivat, P., Perkins, C., and M. Handley, "SDP: Session Description Protocol", RFC 8866, DOI 10.17487/RFC8866, January 2021, <https://www.rfc-editor.org/info/rfc8866>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

Appendix A. Acknowledgements

The authors would like to thank the following for their contributions to the design described in the present document: Brandon Butterworth, Chris Poole, Craig Taylor and David Waring.

We are also grateful to Thomas Swindells and Magnus Westerlund for their helpful review comments.

Appendix B. Examples

This appendix contains examples of multicast QUIC session advertisement and resource transfer (with and without application-layer content security).

B.1. Session Advertisement

This section shows several different examples of an HTTP service advertising a multicast QUIC session. Examples are given in IPv4 form, using reserved address ranges as specified in [RFC5737] and [RFC6676].

B.1.1. Source-specific Multicast QUIC Session

Advertisement of a multicast QUIC session operating on the source-specific multicast group address 232.0.0.1 on port 2000 with the source address 192.0.2.1. The session ID is 16 (0x10) and the idle timeout is one minute. At most 10 resources may be concurrently active in the session and the flow rate should not exceed 10 kbits/s. The multicast transport is unencrypted.

HTTP Alternative Service header field:

```
Alt-Svc:
    h3m="232.0.0.1:2000"; source-address="192.0.2.1";
    session-id=10; session-idle-timeout=60;
    max-concurrent-resources=10; peak-flow-rate=10000
```

B.1.2. Source-specific Multicast QUIC Session with Transport Encryption using a Symmetric Key

Advertisement of a multicast QUIC session operating on the IPv6 globally-scoped source-specific multicast group address ff3e::1234 on port 2000 with the source address 2001:db8::1. The session ID is 16 (0x10) and the idle timeout is one minute. At most 10 resources may be concurrently active in the session and the flow rate should not exceed 10 kbits/s. The multicast transport is encrypted using the AEAD cipher suite 0x13,0x01 (TLS_AES_128_GCM_SHA256) with the shared session key and IV provided.

HTTP Alternative Service header field:

```
Alt-Svc:
    h3m="[ff3e::1234]:2000"; source-address="2001:db8::1";
    session-id=10; session-idle-timeout=60;
    max-concurrent-resources=10; peak-flow-rate=10000;
    cipher-suite=1301; key=4adf1eab9c2a37fd;
    iv=4dbe593acb4d1577ad6ba7dc3189834e
```

B.1.3. Source-specific Multicast QUIC Session with Transport Encryption, Content Integrity and Authenticity

Advertisement of a multicast QUIC session operating on the IPv6 globally-scoped source-specific multicast group address ff3e::1234 on port 2000 with the source address 2001:db8::1. The session ID is 16 (0x10) and the idle timeout is one minute. At most 10 resources may be concurrently active in the session and the flow rate should not exceed 10 kbits/s. The multicast transport is encrypted using the AEAD cipher suite 0x13,0x01 (TLS_AES_128_GCM_SHA256) with the shared session key and IV provided. Content integrity is in use with the digest algorithm set restricted to SHA-256. Content authenticity is in use with the signature algorithm set restricted to rsa-sha256.

HTTP Alternative Service header field:

```
Alt-Svc:
    h3m="[ff3e::1234]:2000"; source-address="2001:db8::1";
    session-id=10; session-idle-timeout=60;
    max-concurrent-resources=10; peak-flow-rate=10000;
    cipher-suite=1301; key=4adf1eab9c2a37fd;
    iv=4dbe593acb4d1577ad6ba7dc3189834e;
    digest-algorithm=SHA-256; signature-algorithm=rsa-sha256
```

B.2. Resource Transfer

This section shows several different examples of the HTTP message patterns for a single resource.

Examples that show HTTP/3 PUSH_PROMISE or HEADERS frames describe the contents of enclosed header block fragments.

B.2.1. Transfer without Content Integrity or Authenticity

HTTP/3 PUSH_PROMISE frame:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
```

HTTP/3 HEADERS frame:

```
:status: 200
content-length: 100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
```

HTTP/3 DATA frame containing 100 bytes of response body data:

```
...
```

B.2.2. Transfer of Partial Content without Content Integrity or Authenticity

In this example, partial content is transferred as described in Section 8. The Range request header is used to indicate the sender's original intention to transfer all 100 bytes of the representation. The Content-Range response header indicates that only the first 50 bytes were actually sent.

HTTP/3 PUSH_PROMISE frame:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
range: bytes=0-*
```

HTTP/3 HEADERS frame:

```
:status: 206
content-length: 100
content-range: bytes 0-49/100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
```

HTTP/3 DATA frame containing 50 bytes of response body data:

```
...
```

B.2.3. Transfer with Content Integrity and without Authenticity

In this example, content integrity is assured by the inclusion of the Digest response header, as described in Section 6.1.

HTTP/3 PUSH_PROMISE frame:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
```

HTTP/3 HEADERS frame including the Digest header:

```
:status: 200
content-length: 100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
digest: SHA-256=DieQ9zfRaDdyAilTVCvmBePuxMm+B6cNocP+QCrNSqo=
```

HTTP/3 DATA frame containing 100 bytes of response body data:

```
...
```

B.2.4. Partial Transfer with Content Integrity and without Authenticity

In this example, partial content is transferred as described in Section 8. The Range request header is used to indicate the sender's intention to transfer all 100 bytes of the representation. The Content-Range response header indicates that only the first 50 bytes were actually sent. Content integrity is assured by the inclusion of the Digest response header, as described in Section 6.1.

HTTP/3 PUSH_PROMISE frame:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
range: bytes=0-*
```

HTTP/3 HEADERS frame including the Digest header:

```
:status: 206
content-length: 100
content-range: bytes 0-49/100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
digest: SHA-256=DieQ9zfRaDdyAilTVCvmBePuxMm+B6cNocP+QCrNSqo=
```

HTTP/3 DATA frame containing 50 bytes of response body data:

```
...
```

B.2.5. Transfer with Content Integrity and Authenticity

In this example, content integrity is assured by the inclusion of the Digest response header, as described in Section 6.1. Content authenticity is assured separately for the request and the response messages by the Signature header which protects the header fields described in further detail below. The Signature header parameter keyId contains the URL of a file containing the public key related to the multicast sender's private key used to create the digital signature.

HTTP/3 PUSH_PROMISE frame including a Signature header protecting the :method and :path (the request target), as well as the :scheme and :authority of the pseudo-request:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
signature: keyId="https://example.org/mcast-sender.example.org.pem",
    algorithm=rsa-sha256,
    headers="(request-target) :scheme :authority",
    signature="MGQCMFTaFptaM2FhgzJq2i9AaChuFDHjp6GiXVtRnI8BsA"
```

HTTP/3 HEADERS frame including a Signature header protecting the :method, :path, :scheme and :authority of the pseudo-request above, plus the Date and Digest of the response:

```
:status: 200
content-length: 100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
digest: SHA-256=DieQ9zfRaDdyAilTVCvmBePuxMm+B6cNocP+QCrNSqo=
signature: keyId="https://example.org/mcast-sender.example.org.pem",
    algorithm=rsa-sha256,
    headers="(request-target) :scheme :authority date digest",
    signature="MGUCMBgx6cuwTzg0W29zNUra8xfMsEcb3rFA3Y"
```

HTTP/3 DATA frame containing response body data:

```
...
```

B.2.6. Partial Transfer with Content Integrity and Authenticity

In this example, partial content is transferred and the Range header (as described in Section 8) is used to indicate that 50 bytes out of 100 bytes were transferred. Content integrity is provided by the inclusion of the Digest header, as described in Section 6.1. Authenticity is provided by the Signature header which protects the header fields described in further detail. The Signature header parameter keyId contains the URL of a file containing the public key related to the multicast sender's private key used to create the digital signature.

HTTP/3 PUSH_PROMISE frame:

```
:method: GET
:scheme: https
:path: /files/example.txt
:authority: example.org
range: bytes=0-*
signature: keyId="https://example.org/mcast-sender.example.org.pem",
    algorithm=rsa-sha256,
    headers="(request-target) :scheme :authority range",
    signature="MGQCMFTaFptaM2FhgzJq2i9AaChuFDHjp6GiXVtRnI8BsA"
```

HTTP/3 HEADERS frame protecting the :method, :path, :scheme and :authority of the pseudo-request above, plus the Date, Digest and Content-Range of the response:

```
:status: 206
content-length: 100
content-range: bytes 0-49/100
content-type: text/plain
date: Fri, 20 Jan 2017 10:00:00 GMT
digest: SHA-256=DieQ9zfRaDdyAilTVCvmBePuxMm+B6cNocP+QCrNSqo=
signature: keyId="https://example.org/mcast-sender.example.org.pem",
    algorithm=rsa-sha256,
    headers="(request-target) :scheme :authority
        date digest content-range",
    signature="MGUCMBgx6cuwTzg0W29zNUra8xfMsEcb3rFA3Y"
```

HTTP/3 DATA frame containing response body data:

```
...
```

Appendix C. Summary of differences from unicast QUIC and HTTP/3

Protocol feature	Unicast QUIC	Multicast QUIC profile
Packet number spaces	QUIC transport packets are seperated into three distinct packet number spaces: initial, handshake and application data.	All packets are numbered in the application data packet number space.
Transport handshake	Combined cryptographic and transport handshake stream conveying TLS handshake messages in QUIC Initial and Handshake packets.	Not used.
TLS cipher suite	Client's preferred cipher suite included in Client Hello message.	Cipher suite optionally advertised out of band via Alt-Svc cipher-suite parameter. Default cipher suite is 0x00,0x00 (NULL_WITH_NULL_NULL).
TLS session key	Session key included in Server Hello message.	Session key optionally advertised out of band via Alt-Svc key parameter.
TLS initialization vector	Initialization vector included in Server Hello message.	Initialization vector optionally advertised out of band via Alt-Svc iv parameter.

Table 1: Cryptography and Transport

Protocol feature	Unicast QUIC	Multicast QUIC profile
initial_max_data	Connection-level flow control window size.	Not used. Peak flow rate of a session optionally advertised out of band via Alt-Svc peak-flow-rate parameter.
initial_max_stream_data_bidi_local	Locally-initiated bidirectional stream flow control window size.	Not used. No bidirectional streams in this profile.
initial_max_stream_data_bidi_remote		

Protocol feature	Unicast QUIC	Multicast QUIC profile
	Remotely-initiated bidirectional stream flow control window size.	Not used. No bidirectional streams in this profile.
initial_max_stream_data_uni	Unidirectional stream flow control window size.	Not used. Peak flow rate of a session optionally advertised out of band via Alt-Svc peak-flow-rate parameter.
initial_max_streams_bidi and initial_max_streams_uni	Maximum stream concurrency.	Not used. Maximum concurrent resources in a session optionally advertised out of band via Alt-Svc max-concurrent-resources parameter.

Table 2: Required Transport Parameters

Protocol feature	Unicast QUIC	Multicast QUIC profile
original_destination_connection_id	The value of the Destination Connection ID field from the first Initial packet sent by the client.	Not used. No client interaction.
max_idle_timeout	How long to keep an idle connection open for before closing. Takes a default of 0 (never close on idle) if not specified.	Not used. Advertised out of band via Alt-Svc session-idle-timeout parameter; defaults to 0

Protocol feature	Unicast QUIC	Multicast QUIC profile
		(never close on idle) if not specified.
stateless_reset_token	Used in verifying a stateless reset.	Not used. Stateless reset is not used by this profile.
max_udp_payload_size	Limit of the size of packets that an endpoint is willing to receive.	Not used. Maximum packet size for a session optionally advertised out of band via Alt-Svc max-packet-size parameter.
ack_delay_exponent	The exponent used to decode the ACK Delay field in the ACK frame.	Not used. ACK frames are prohibited by this profile.
max_ack_delay	Maximum time in milliseconds by which an endpoint will delay sending acknowledgements.	Not used. ACK frames are prohibited by this profile.
disable_active_migration	Signals if an endpoint does not support connection migration.	Not used. Session migration not currently supported by this profile.
preferred_address	Used to effect a change in server address at the end of the handshake.	Not used. No handshake in this profile.
active_connection_id_limit		Not used. Only a single session identifier is used in this profile.
initial_source_connection_id		

Protocol feature	Unicast QUIC	Multicast QUIC profile
	The value that an endpoint included in the Source Connection ID field of the first Initial packet it sent for the connection	Not used. No client interaction.
retry_source_connection_id	The value that the server included in the Source Connection ID field of a retry packet	Not used. Retry packets are not used in this profile.

Table 3: Optional Transport Parameters

Protocol feature	Unicast QUIC	Multicast QUIC profile
Maximum packet size	Determined by path MTU discovery or other heuristic.	Determined by path MTU discovery or other heuristic.
Long header packet	Used for packets that are sent prior to the completion of version negotiation and before packet protection keys are established.	Prohibited.
Version negotiation packet	Protocol version negotiation between initiating client and server.	Not permitted.
Stateless reset packet	Used by a peer to terminate a connection that has become unusable.	Not permitted. (Potential denial-of-service attack vector.)
Short header packet	Used for packets that are sent once a connection has been established. Used to convey QUIC frames (see below).	Used to convey QUIC frames (see below).
Source Connection ID field, Destination Connection ID field	Connection IDs negotiated between client and server during handshake and may be changed subsequently using the NEW_CONNECTION_ID frame.	Source Connection ID not used. Destination Connection ID redefined to be a Multicast Session ID which is optionally advertised out of band via the Alt-Svc session-id

Protocol feature	Unicast QUIC	Multicast QUIC profile
		parameter. May be omitted from packets if the address/port tuple is sufficient to identify the session, in which case it is not advertised.

Table 4: QUIC Packet Layer

Protocol feature	Unicast QUIC	Multicast QUIC profile
PADDING frame	Used to pad out a QUIC packet with zero bytes. (The first packet sent on a connection must be at least 1200 bytes long to prove that the network can transmit packets of at least this size.)	Permitted, but wasteful of network capacity.
PING frame	Used by an endpoint to verify that its peer is still alive. Acknowledged using a regular ACK frame.	Used by the multicast sender as a session keep-alive notification. Never acknowledged.
ACK frames	Used by a receiver to acknowledge receipt of data from its peer.	Both ACK frame types are prohibited.
RESET_STREAM frame	Request by receiver for sender to terminate a stream transmission prematurely.	Indication to receivers that the multicast sender has prematurely terminated a stream transmission. Prohibited for receivers to send.
STOP_SENDING frame	Flow control back pressure. Used to signal to a peer that incoming data on a stream is being discarded on receipt at the application's request. This abruptly terminates a stream.	Prohibited.
CRYPTO frame	Used to transmit cryptographic handshake messages.	Prohibited.
NEW_TOKEN frame		

Protocol feature	Unicast QUIC	Multicast QUIC profile
	Used by a server to supply a token to its client to be sent in the header of an initial packet for a future connection. Used to facilitate connection migration.	Prohibited. Session migration not currently supported by this profile.
STREAM frame	Conveys application-layer payloads (see HTTP/3 mapping below).	Conveys application-layer payloads (see HTTP/3 mapping below).
FIN bit on STREAM frame	Indication by sender to its peer that a stream transmission has finished.	Indication by the multicast sender that a stream transmission has finished.
MAX_DATA frame	Flow control update notification.	Prohibited.
MAX_STREAM_DATA frame	Flow control update notification.	Prohibited.
MAX_STREAMS frame	Used by an endpoint to inform a peer of the maximum stream ID that it is permitted to open.	Prohibited.
DATA_BLOCKED frame	Notification to receiver that sender's transmission is blocked pending an update to its flow control window by peer.	Prohibited.
STREAM_DATA_BLOCKED frame	Notification to receiver that sender's transmission of a single stream is blocked pending an update to its flow control window by peer.	Prohibited.
STREAMS_BLOCKED frames	Notification to receiver that the sender wishes to open a stream but is unable to do so because the maximum stream ID has been reached for a given stream type.	Prohibited.
NEW_CONNECTION_ID frame	Used to provide a peer with alternative	Prohibited. Session migration not

Protocol feature	Unicast QUIC	Multicast QUIC profile
	connection IDs that can be used to break linkability when migrating connections.	currently supported by this profile.
RETIRE_CONNECTION_ID frame	Indicates that an endpoint will no longer use a connection ID that was issued by its peer.	Prohibited. Session migration not currently supported by this profile.
PATH_CHALLENGE frame	Used to check reachability to a peer and for path validation during connection migration.	Prohibited.
PATH_RESPONSE frame	Sent in response to a PATH_CHALLENGE frame.	Prohibited.
CONNECTION_CLOSE frame	Notification (by either peer) of graceful connection shutdown.	Prohibited. Use HTTP explicit session tear-down instead (see Section 5.4).
HANDSHAKE_DONE frame	Used by a server to inform a client that the cryptographic handshake has completed.	Prohibited.

Table 5: QUIC Framing Layer

Protocol feature	Unicast HTTP/3	Multicast HTTP/3 profile
Stream Type	Type of unidirectional stream.	Only Server Push type is permitted.
Push ID	Identifies a promised resource conveyed in an earlier PUSH_PROMISE frame.	Identifies a promised resource conveyed in an earlier PUSH_PROMISE frame.
DATA frame	Carriage of HTTP request/response message body.	Carriage of HTTP response message body.
HEADERS frame	Carriage of HTTP request/response message metadata. Trailing HEADERS frame is permitted.	Carriage of HTTP response message metadata. Trailing HEADERS frame is permitted.
CANCEL_PUSH frame	Used to request cancellation of server push prior to the push stream being created.	Permitted only for senders.

Protocol feature	Unicast HTTP/3	Multicast HTTP/3 profile
SETTINGS frame	Negotiation of HTTP/3 connection settings.	Prohibited.
PUSH_PROMISE frame	Carriage of HTTP pseudo-request message metadata.	Carriage of HTTP pseudo-request message metadata. All HTTP representation transfers over multicast begin this way. Stream ID of the first client-initiated bidirectional stream is reserved and all PUSH_PROMISE frames reference this as the client request from which they are notionally derived. This stream remains open while a sender is participating in the multicast QUIC session.
GOAWAY frame	Early notification (by either endpoint) of future connection closure, allowing for orderly completion of open streams.	Prohibited. Use HTTP explicit session tear-down instead.
MAX_PUSH_ID frame	Used by a receiver to control the number of server pushes that a sender can initiate.	Prohibited.
ALTSVC HTTP/2 extension frame	Signalling alternative service for HTTP/3 session.	Prohibited.
Other HTTP/2 extension frames		Prohibited.

Table 6: HTTP/3 Mapping

Protocol feature	Unicast HTTP/3	Multicast HTTP/3 profile
Huffman string compression	Header blocks may use Huffman codes to compress literal string values.	Header blocks may use Huffman codes to compress literal string values.
Static table	Header blocks may refer to indexed entries in the static table.	Header blocks may refer to indexed entries in the static table.
Dynamic table	Header blocks may insert entries into the dynamic table and	

Protocol feature	Unicast HTTP/3	Multicast HTTP/3 profile
	subsequent header blocks may refer to their indexes. Unused as size is currently locked to 0.	Prohibited, size is currently locked to 0.

Table 7: HTTP Metadata Compression Layer

Appendix D. Changelog

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

D.1. Since draft-pardue-quic-http-mcast-10

*Remove stale Author's Notes about QUIC and HTTP/3 specifications being in flux now that the specifications are RFCs

D.2. Since draft-pardue-quic-http-mcast-09

*Update references to HTTP/3 and QPACK I-Ds to [RFC9114] and [RFC9204] respectively.

*Update reference to DATA_WITH_OFFSET frame draft.

D.3. Since draft-pardue-quic-http-mcast-08

*Update references to QUIC WG I-Ds to [RFC9000], [RFC9001] and [RFC9002].

*Update reference to DATA_WITH_OFFSET frame draft.

*Update informative reference for SDP to [RFC8866] as 4566 has been obsoleted by it

D.4. Since draft-pardue-quic-http-mcast-07

*Update references to QUIC WG I-Ds.

*Remove stale references to sections of the QUIC WG I-Ds that don't exist any more.

*Add references to the DATA_WITH_OFFSET frame I-D.

D.5. Since draft-pardue-quic-http-mcast-06

*Update references to QUIC WG I-Ds.

*Clarify that only the first source-address parameter should be used if duplicated.

*Fix source-address example to not be a quoted string.

*Fix bytes in the ALPN identification sequence following change to h3m.

*Update language around QUIC Connection IDs to reflect the core specs.

*Update frame and transport parameter names throughout to match core specifications.

*Remove Author's Note about reserved stream ID for PUSH_PROMISE frames.

*Update language around QPACK static and dynamic table indexing to more closely align with the core spec.

*Update Session Idle Timeout to match core specs, including removing limitation of 600 seconds and expressing in milliseconds, not seconds.

*Preface Session Termination with a statement clarifying that participants may leave at any time.

D.6. Since draft-pardue-quic-http-mcast-05

*Update references to QUIC WG I-Ds.

*Sender packet number size is now fixed for the duration of a session.

*Change how to handle multiple session IDs: sessions are now only allowed a single ID.

*Remove incompatible requirements set by [RFC9000]'s "Required Operations".

*Additionally ban HANDSHAKE_DONE.

*Remove Version Negotiation now that the quic Alt-Svc parameter has been removed (examples also updated).

*Remove HTTP Prioritization references.

*Add new extensions Alt-Svc parameter.

*Broaden peak flow rate to QUIC payload to encompass all frame types.

*Change ALPN identifier to h3m.

D.7. Since draft-pardue-quic-http-mcast-04

*Update references to QUIC WG I-Ds, remove QUIC-SPIN. (draft-ietf-quic-transport-20)

*Update session ID length to match new connection ID length. (draft-ietf-quic-transport-22)

*Clarify the mapping for the new active_connection_id_limit session parameter. (draft-ietf-quic-transport-21)

*Update text to conform with latest version negotiation text from [RFC9000].

*Update stream types for server push in accordance with draft-ietf-quic-http-19.

*Fix some idnits warnings to do with line lengths in Alt-Svc examples.

*Update IPv6 informative reference to RFC 8200 as it obsoletes RFC 2460.

*Clarify difference between connection and session migration.

*Move GOAWAY frame to HTTP/3 profile.

*Renamed Session Shutdown to Connection Shutdown to mirror concept in [RFC9000].

*Clarify the layer of each frame type when referred to.

D.8. Since draft-pardue-quic-http-mcast-03

*Update references to QUIC WG I-Ds.

*Change crypto handshake text now that it's no longer done on Stream ID 0.

*Update to reference Source and Destination Connection IDs.

*Prohibit the use of connection coalescing, migration and ECN.

*Update allowed and disallowed packets and frames to match the core specs.

*Remove references to the PONG frame. (draft-ietf-quic-transport-10)

*Change HTTP/QUIC to HTTP/3. (draft-ietf-quic-http-17)

*Change HPACK to QPACK. (draft-ietf-quic-http-10)

*Prohibit the DUPLICATE_PUSH frame.

*Clarify packet number space (only use application data space, not initial or handshake).

*Add statement on QUIC latency spin bit.

*Removed sentence stating that multiple Connection IDs cannot be used concurrently in a unicast QUIC session, in accordance with [RFC9000] section 5.1.2.

D.9. Since draft-pardue-quic-http-mcast-02

*No changes.

D.10. Since draft-pardue-quic-http-mcast-01

*Explicit guidance on maximum stream ID value permitted.

*Updated guidance on PING (and PONG) frame.

*Added a comparison table to appendix.

*Remove invalid use of trailing headers.

*Use the new HTTP/QUIC DATA frame.

*Prohibit APPLICATION CLOSE frame, and move GOAWAY frame to HTTP/QUIC section.

*Redefine server push to reflect core document changes.

*Remove default idle time out value.

*Clarify session parameter requirements (session-idle-timeout became mandatory).

*Update frame notation convention.

D.11. Since draft-pardue-quic-http-mcast-00

*Update references to QUIC WG I-Ds.

*Relax session leaving requirements language.

*Clarify handling of omitted session parameter advertisements.

*Rename "Idle" state to "Quiescent".

*Add digest algorithm session parameter.

*Add signature algorithm session parameter.

*Add Initialization Vector session parameter.

*Replace COPT tag-value-pairs with TransportParameter values.

*Add example of an advertisement for a session with content authenticity and integrity.

Authors' Addresses

Lucas Pardue

Email: lucaspardue.24.7@gmail.com

Richard Bradbury
BBC Research & Development

Email: richard.bradbury@bbc.co.uk

Sam Hurst
BBC Research & Development

Email: sam.hurst@bbc.co.uk