moq
Internet-Draft
Intended status: Informational
Expires: 18 September 2026

L. Curley

17 March 2026

MoQ Probe Extension
draft-lcurley-moq-probe-00

Abstract

This document defines a PROBE extension for MoQ Transport [moqt]. A subscriber opens a bidirectional PROBE stream to request that the publisher pad the connection up to a target bitrate. The publisher sends padding as defined in [moqt] Section 7.7 and periodically responds with the measured bitrate and an elapsed timestamp, enabling the subscriber to estimate the available bandwidth.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Media Over QUIC Working Group mailing list (moq@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/moq/.

Source for this draft and an issue tracker can be found at https://github.com/kixelated/moq-drafts.

Status of This Memo

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

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

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

This Internet-Draft will expire on 18 September 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Conventions and Definitions

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 capitals, as shown here.

2. Introduction

Bandwidth estimation is essential for adaptive bitrate media delivery. A subscriber needs to know the available bandwidth in order to select appropriate media tracks and qualities.

2.1. Application-Limited

Many MoQ applications are application-limited: the average bitrate of the media is less than the available bandwidth. Most congestion control algorithms only grow the congestion window or bandwidth estimate when fully utilized. This means the available bandwidth is often underestimated, and the subscriber has no way to know if it can safely switch to a higher quality track.

This is particularly problematic for adaptive bitrate (ABR) algorithms. A viewer may get stuck at a low quality rendition indefinitely because the congestion window never grows to reflect the true link capacity. If the viewer does attempt to switch to a higher rendition without first probing, they risk buffering — either because the congestion window has not been warmed up to support the higher bitrate, or because the network genuinely cannot sustain it. Without probing, the subscriber cannot distinguish between these two cases.

[moqt] Section 3.7.2 suggests subscribing to additional tracks at low priority or sending padding to fill the congestion window during probing intervals. However, this is difficult in practice because the subscriber does not know when probing is needed or by how much. The congestion window and bandwidth estimate are internal to the sender’s congestion controller and are not exposed to the application, let alone the remote peer. The subscriber cannot distinguish between "the network has more capacity" and "the congestion controller is already fully utilizing the link".

2.2. Hop-by-Hop

MoQ is designed to work end-to-end via relays. Each hop may have different network conditions, so bandwidth estimation must be performed per-hop rather than end-to-end. A subscriber needs to know the capacity of its immediate connection, not the capacity of the origin.

Using a wire-level extension ensures that PROBE measurements are scoped to a single hop. A relay terminates the PROBE stream and does not forward it upstream, avoiding incorrect measurements that reflect intermediate link capacity.

2.3. This Extension

[moqt] defines padding streams and datagrams (Section 7.7) for probing, but does not define a signaling mechanism for a subscriber to request padding or for a publisher to report measurements. This extension fills that gap: the subscriber opens a PROBE stream to request that the publisher pad the connection to a target bitrate using [moqt] padding. The publisher responds with periodic measurements, allowing the subscriber to adjust its subscriptions accordingly.

3. Setup Negotiation

The PROBE extension is negotiated during the SETUP exchange as defined in [moqt] Section 9.4.

Both endpoints indicate support by including the following Setup Option:

```
PROBE Setup Option {
  Option Key (vi64) = TBD1
  Option Value Length (vi64) = 0
}
```

If both endpoints include this option, the PROBE extension is available for the session. If a peer receives a PROBE stream without having negotiated the extension, it MUST close the session with a PROTOCOL_VIOLATION.

4. PROBE Stream

The PROBE extension uses a new bidirectional stream type.

STREAM_TYPE = TBD2

The stream type is sent at the beginning of the stream, encoded as a variable-length integer, consistent with [moqt] stream type framing.

A subscriber (stream opener) sends PROBE_REQUEST messages on the stream. The publisher (responder) sends PROBE_RESPONSE messages on the stream. Either endpoint MAY close or reset the stream at any time.

4.1. PROBE_REQUEST

A subscriber sends a PROBE_REQUEST to indicate the target the publisher should attempt to reach.

```
PROBE_REQUEST {
  Message Length (vi64)
  Target Bitrate (vi64)
}
```

Target Bitrate: The desired bitrate in kilobits per second. The publisher SHOULD send padding as defined in [moqt] Section 7.7 to attempt to reach this rate. A value of 0 indicates no padding is needed; the publisher SHOULD only send media data but MUST continue sending PROBE_RESPONSE messages. This is useful for passively monitoring the current bitrate without actively probing for more bandwidth. Either endpoint MAY close or reset the stream to stop receiving updates entirely.

The subscriber MAY send multiple PROBE_REQUEST messages on the same stream. Each new PROBE_REQUEST supersedes the previous one. The publisher MUST use the most recently received target.

4.2. PROBE_RESPONSE

The publisher periodically sends PROBE_RESPONSE messages containing the measured bitrate and the elapsed time since the last response.

```
PROBE_RESPONSE {
  Message Length (vi64)
  Measured Bitrate (vi64)
  Elapsed (vi64)
}
```

Measured Bitrate: The estimated bitrate in kilobits per second. How this value is computed is implementation-defined and depends on the congestion controller. Pacing-based algorithms (e.g. BBR) can report the current pacing rate directly, while window-based algorithms (e.g. CUBIC, Reno) may want to smooth the estimate since the sending rate is inherently bursty. This includes media, padding, and any other data sent by the publisher.

Elapsed: The number of milliseconds since the previous PROBE_RESPONSE on this stream. For the first PROBE_RESPONSE, this is the number of milliseconds since the corresponding PROBE_REQUEST was received. This allows the subscriber to assess the freshness of the measurement and detect stale updates caused by network delays.

The publisher SHOULD send PROBE_RESPONSE messages at regular intervals while probing is active. The interval is implementation-defined but a value between 100ms and 1000ms is RECOMMENDED.

5. Padding

The publisher SHOULD send padding using the mechanisms defined in [moqt] Section 7.7 (padding streams and/or padding datagrams).

The publisher MUST NOT exceed the target with padding alone. If media traffic already meets or exceeds the target, no additional padding is necessary.

The publisher MUST respect the QUIC congestion controller. Padding that would cause the congestion window to be exceeded MUST NOT be sent. The goal is to fill unused capacity, not to cause congestion.

6. Security Considerations

A malicious subscriber could request an excessively high target to waste publisher resources or cause network congestion. Implementations SHOULD enforce reasonable limits on the target and MAY ignore or cap requests that exceed these limits.

A publisher SHOULD rate-limit the amount of padding it sends to avoid being used as an amplification vector.

A publisher MAY rate-limit or ignore frequent PROBE_REQUEST messages to prevent flooding or oscillation. Implementations SHOULD enforce a minimum inter-request interval for PROBE_REQUESTs from a given subscriber.

7. IANA Considerations

This document requests the following registrations:

7.1. MOQT Setup Option Type

This document registers the following entry in the "MoQ Setup Option Types" registry:

Value	Name	Reference
TBD1	PROBE	This Document

Table 1

7.2. MOQT Stream Type

This document registers the following entry in the "MoQ Stream Types" registry:

Value	Name	Reference
TBD2	PROBE	This Document

Table 2

8. Normative References

[moqt] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-17>.

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

Acknowledgments

This document was drafted with the assistance of Claude, an AI assistant by Anthropic.

Author’s Address

Luke Curley
Email: kixelated@gmail.com