Media Over QUIC
Internet-Draft
Intended status: Standards Track
Expires: 21 May 2026

T. Evens
T. Rigaux
Cisco
S. Henning

17 November 2025

Media over QUIC Relay Benchmark Methodology
draft-evens-moq-bench-00

Abstract

This document defines a comprehensive methodology for benchmarking Media over QUIC Transport (MOQT) relays to evaluate their performance under realistic conditions. The methodology utilizes configurable test profiles that simulate common media streaming scenarios including audio-only distribution, combined audio-video streaming, and multi-participant conferencing environments. The framework defines standardized message formats, synchronization mechanisms, and comprehensive metrics collection to ensure reproducible and comparable results across different relay implementations. Test scenarios cover both (QUIC) datagram and stream forwarding modes, enabling evaluation of relay performance characteristics such as throughput, latency variance, object loss rates, and resource utilization. The resulting benchmark data facilitates objective comparison and analysis of (MOQT) relay implementations, supporting informed deployment decisions and performance optimization efforts.

About This Document

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

The latest revision of this draft can be found at https://timevens.github.io/draft-evens-moq-bench/draft-evens-moq-bench.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-evens-moq-bench/.

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

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. Data Patterns 5
  3.1. Video Media 5
  3.2. Audio Media 5
  3.3. Fetch 6
  3.4. Interactive Data 6
  3.5. Group and Subgroup Impact to Relays 6
4. Scenarios 6
  4.1. Scenario 1: Single Publisher to Multiple Subscribers - Audio 6
  4.2. Scenario 2: Single Publisher to Multiple Subscribers - Audio and Video 7
  4.3. Scenario 3: Groups of Publishers and Subscribers (aka Meeting) 8

5. Methodology 8
5.1. Config Profile 8
5.2. Synchronize Start of Benchmark 10
5.3. Completion of benchmark 10
5.4. Messages 11
5.4.1. START 11
5.4.2. DATA 11
5.4.3. COMPLETION 11
5.5. Benchmark Metrics 12
5.5.1. While Track is in Progress 12
5.5.2. On completion of track 12
5.6. Running Benchmark 13
5.7. Test Environment Isolation 14
5.8. Authentication and Authorization 14
5.9. Resource Management 14
5.10. Data Privacy 14
5.11. Implementation Security 14
6. IANA Considerations 15
7. Security Considerations 15
8. References 15
8.1. Normative References 15
8.2. Informative References 15
Appendix A. Example Benchmark Results 16
A.1. Environment 16
A.2. Benchmark Tool 16
A.3. Relays 16
A.4. Scenario 1: Single publisher audio 16
A.4.1. Config Profile 16
A.4.2. Results 17
A.5. Scenario 2: Single publisher audio/video 17
A.5.1. Config Profile 17
A.5.2. Results 18
Acknowledgments 19
Authors' Addresses 19

1. Introduction

[MOQT] specifies the client-server interactions and messaging used by clients and relays to fan-out published data to one or more subscribers. Implementations employ several state machines to coordinate these interactions. Relays must serialize and deserialize data objects and their extension headers when forwarding to subscribers. Because [MOQT] runs over [QUIC], relay performance is directly affected by receive/send processing, encryption/decryption, and loss-recovery ACK handling.

To evaluate relay performance under realistic conditions, this document defines a benchmarking methodology that mimics real-world use cases.

2. Terminology

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.

Commonly used terms in this document are described below.

Track Configurations: Track configurations define the parameters and settings for individual media tracks used in benchmarking, including transmission patterns, object sizes, intervals, and delivery modes.

Config Profile: A set of Track Configurations that are used to perform a series of tests to evaluate the performance of relays.

Test Scenario: A test scenario is a set of config profiles that are used to perform a series of tests to evaluate the performance of relays.

Relay: A [MOQT] relay is an intermediary that receives published media objects from publishers and forwards them to subscribers, providing fan-out distribution capabilities.

Publisher: An entity that produces and sends media objects to a [MOQT] relay for distribution to subscribers.

Subscriber: An entity that receives media objects from a [MOQT] relay by subscribing to specific tracks.

Track: A sequence of related media objects identified by a namespace and track name, representing a single media stream such as audio or video.

Group: A collection of related objects within a track that have dependencies on each other, typically used for video frames where objects within a group are interdependent.

Object: An individual unit of media data transmitted within a track, such as an audio sample or video frame.

Namespace: A hierarchical identifier that groups related tracks

together, allowing for organized distribution of media content.

Datagrams: [QUIC] datagram frames used for unreliable, low-latency transmission of small media objects, typically audio.

Streams: [QUIC] streams used for reliable, ordered transmission of media objects, typically video or larger content.

3. Data Patterns

Data transmission patterns are relative to the data that is being transmitted at the time of transmission.

3.1. Video Media

Video media is a common [MOQT] use case. A video media track typically begins with a large data object, followed by a series of smaller data objects within the same group. The initial object is often significantly larger than the subsequent ones. Video is transmitted at regular intervals (e.g., every 33 ms for 30 fps), producing a bursty fan-out pattern at each interval. Each data object commonly carries at least three extension headers.

Video has a direct dependency on previous objects within a group. If objects are lost in a group, it breaks the ability to render the video correctly.

Video is often larger than MTU and cannot be transmitted as defined in [MOQT] using datagram. Unless video is very low quality, video uses [QUIC] stream forwarding instead of datagram forwarding.

3.2. Audio Media

Audio media is a common [MOQT] use case. An audio media track typically maintains fixed intervals with similarly sized data objects. Like video, transmission occurs at each interval, producing a periodic fan-out burst. Each data object usually carries fewer than three extension headers.

[MOQT] does not define how to handle larger than MTU sized data objects for [QUIC] datagrams. Audio does not have the same requirement as video where subsequent data objects are related to the previous object or group.

Audio is a prime candidate for datagram considering the size of the data objects are less than MTU and there is no dependency on previous object within a group.

Audio primarily uses [QUIC] datagrams for forwarding.

3.3. Fetch

A fetch in [MOQT] is similar to a retrieval of a file where the data will be sent all at once and not paced on an interval. In [MOQT] the data will be sent using a single [QUIC] stream. The data will be sent as fast as the receiver [QUIC] connection can receive it.

3.4. Interactive Data

Chat, metrics, and logging are use cases for interactive data. This data is sent based on interaction, such as a human typing a message or metrics event, or logging event. A unique characteristic of this data is that it is sent when needed and isn’t always constant. Metrics might be an exception where metrics are sent at a constant interval, but metrics may also be sent based on a triggered event.

Interactive data is often sent using [QUIC] streams but can also be sent via datagram considering the size of the data is often less than MTU size.

3.5. Group and Subgroup Impact to Relays

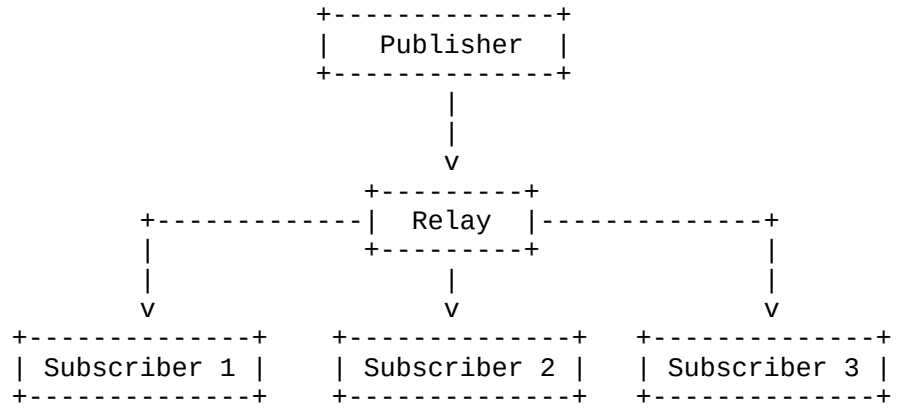
In [MOQT], when a group or subgroup changes a [QUIC] stream is created for the group/subgroup. This results in additional state for the [MOQT] relay implementation to handle multiple groups/subgroups in-flight. The data transmission use case drives how frequently groups/subgroups change, which can impact the performance of relays. Changing of group/subgroup is only impactful with [MOQT] stream tracks.

4. Scenarios

A combination of data patterns are used to define a scenario. The scenario becomes a Config Profile. Various data patterns are combined to define a scenario. The scenario (Config Profile) is used to benchmark a relay.

Below describes common scenarios that the benchmark and configuration profiles need to support. Other scenarios can be defined as needed using the configuration profiles.

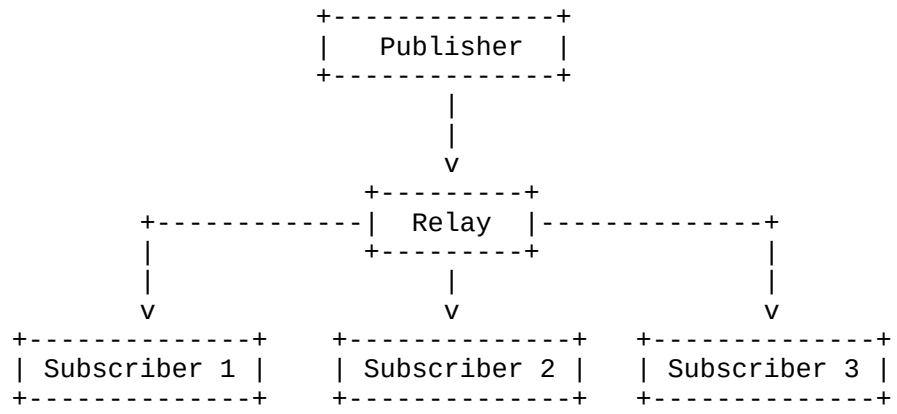
4.1. Scenario 1: Single Publisher to Multiple Subscribers - Audio



The above diagram shows a single publisher sending audio to multiple subscribers.

A single audio track is published using datagram. The audio track is sent at a constant interval that is then sent to all subscribers. The benchmark is to see how many subscribers can be supported by the relay using datagram forwarding.

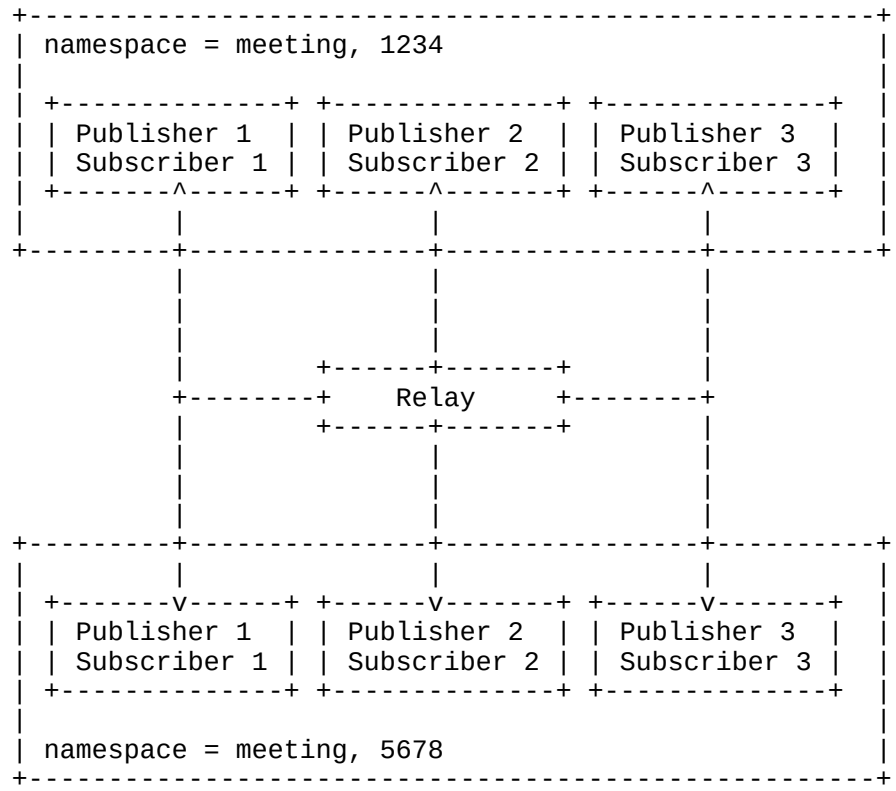
4.2. Scenario 2: Single Publisher to Multiple Subscribers - Audio and Video



The above diagram shows a single publisher sending audio and video to multiple subscribers.

Two tracks are published, an audio datagram track and a reliable stream track for video. The benchmark is to see how many subscribers can be supported by the relay using both datagram and stream forwarding tracks.

4.3. Scenario 3: Groups of Publishers and Subscribers (aka Meeting)



The above diagram shows a set of publishers and subscribers. Each client in this scenario publishes two tracks, audio and video, and subscribe to the other subscribers audio and video tracks. In this scenario, the set of publishers and subscribers mimic a video conference meeting. The benchmark is to see how many sets (aka meetings) can be supported by the relay using datagram and stream forwarding tracks.

5. Methodology

5.1. Config Profile

A configuration profile consists of per track configuration settings. The following track settings are defined:

Field	Notes
namespace	Namespace tuple of the track
name	Name of the track
reliable	True to use QUIC streams and false to use datagrams
mode	Is a value of 1 to indicate publisher, 2 to indicate subscriber, or 3 to indicate both. When both, the track namespace or name will often be interpolated based on the test script
priority	Priority of the track
ttl	Time to live for objects published
interval	Interval in milliseconds to send data objects
Objects Per Group	Number of objects within a group
First Object Size	First object of group size in bytes
Remaining Object Size	Remaining objects of group size in bytes
Duration	Duration in milliseconds to send data objects
Start Delay	Start delay in milliseconds

Table 1: Textual NodeId Formats

Interpolation can be used to template the configuration profile.

Upon start, the track will start off with "First Object Size" of data bytes. Subsequent objects will use the Remaining Object Size in bytes. The remaining objects sent will be Objects Per Group number minus the first object. For example, if Objects Per Group is 10 and First Object Size is 100 bytes and Remaining Object Size is 50 bytes, then the first group will be 100 bytes and the remaining 9 groups will be 50 bytes each.

Groups value will increment based on sending Objects Per Group number of objects. TTL is used to define how long published objects should remain in the transmit queue.

Objects will be published at the Interval value.

Objects and groups will continue for as long as the Duration value.

5.2. Synchronize Start of Benchmark

Synchronization of the benchmark is required to ensure that all clients start at the same time as the publisher so that objects published can be compared to objects received.

To synchronize the start of the benchmark, the publisher will publish a start message (Section 5.4.1) repeatedly for the duration of Start Delay value in milliseconds. The start message will be published every every 1/10th interval of the Start Delay value in milliseconds. If 1/10th is less than 100ms, the start message will be published every 100ms.

Clients will subscribe to the published track and will wait for the start message (Section 5.4.1) to be received. If the start message is not received before data messages or completion message, the track benchmark is considered failed.

Clients will ignore the repeated start messages after receiving the first one. The test begins on the first data message (Section 5.4.2) received. Start message MUST NOT be sent after sending the first data message message.

5.3. Completion of benchmark

The per track benchmark is completed when the publisher sends a completion message (Section 5.4.3). The completion message will be sent after the last data message (Section 5.4.2) message is sent.

5.4. Messages

Testing utilizes various messages to start testing, send data, and report metrics on completion.

The following messages are defined:

5.4.1. START

```
START Message {
  type (uint8) = 0x01,
  objects_per_group (uint32),
  first_object_size (uint32),
  remaining_object_size (uint32),
  interval (uint32),
}
```

5.4.2. DATA

```
DATA Message {
  type (uint8) = 0x02,
  group_number (uint64),
  object_number (uint64),
  milliseconds_since_first_object (uint32_t),
  data_length (uint32),
  data (bytes),
}
```

milliseconds_since_first_object: Number of milliseconds since the first object was sent. The first object starts at zero and is incremented by milliseconds from the publisher for each message sent. The publisher adds the time when it sends the message. If there are publisher delays, then this time would reflect the delay variance from expected interval.

5.4.3. COMPLETION

```
COMPLETION Message {
  type (uint8) = 0x03,
  objects_sent (uint64),
  groups_sent (uint64),
  total_duration (uint32),
}
```

total_duration: Total duration is the total duration in milliseconds from first object to last object sent

5.5. Benchmark Metrics

5.5.1. While Track is in Progress

The following metrics are computed ongoing as the track is in progress:

Metric	Notes
Received Objects	Count of received objects
Received Groups	Count of received groups
Receive Delta	Delta time in milliseconds from last message to current message received
Average Delta	Average delta time in milliseconds from last message to current message received
Max Delta	Maximum delta time in milliseconds seen
Publisher Variance	Received milliseconds_since_first_object should be zero based on interval. This metric tracks the variance of publisher sending delay
Receive Variance	Received milliseconds_since_first_object should be zero based on interval. This metric tracks the variance of received objects based on expected interval
Average Bits Per Second	Average bits per second received

Table 2

5.5.2. On completion of track

The following result metrics are computed at the end of a track that has completed. Each track has the following metrics:

Metric	Notes
Objects Sent	Number of objects reported sent by the publisher
Groups sent	Number of groups reported sent by the publisher
Objects Received	Number of objects received
Groups Received	Number of groups received
Lost Objects	Computed based on objects received to objects sent
Total Duration	Total duration in milliseconds from the publisher
Actual Duration	Duration in milliseconds from received first object to last object received
Average Receive Variance	The average receive variance of received objects based on expected interval
Average Publisher Variance	The average publisher variance of objects published
Average Bits Per Second	The average bits per second received
Expected Bits Per Second	Computed bits per second rate based on Section 5.4.1 first, remaining sizes and interval values

Table 3

5.6. Running Benchmark

Using a configuration profile, multiple instances of clients are ran to connect to the relay. They will publish and subscribe based on the configuration profile. Interpolation is used to dynamically modify the configuration profile namespace and name based on the test script. In the case of mode 3 (both), the namespace and name will be interpolated based on the test script to avoid multi-publisher and mirroring where the publisher subscribes to itself.

5.7. Test Environment Isolation

Benchmarking should be conducted in isolated test environments to prevent:

* Interference with production traffic
* Exposure of test data patterns to unauthorized parties
* Resource exhaustion attacks on production systems

5.8. Authentication and Authorization

Benchmark implementations SHOULD implement proper authentication and authorization mechanisms to:

* Prevent unauthorized participation in benchmark tests
* Ensure only authorized entities can initiate benchmark sessions
* Protect against malicious clients that could skew results

5.9. Resource Management

Benchmark implementations MUST include safeguards to prevent:

* Excessive resource consumption that could lead to denial of service
* Memory exhaustion from unbounded metric collection
* CPU exhaustion from processing malformed benchmark messages

5.10. Data Privacy

While benchmark data is typically synthetic, implementations SHOULD:

* Avoid including sensitive information in test data
* Provide mechanisms to sanitize benchmark results before sharing
* Consider the privacy implications of detailed performance metrics

5.11. Implementation Security

Benchmark tools MUST follow secure coding practices and SHOULD undergo security review, as they may be deployed in sensitive network environments.

6. IANA Considerations

This document does not require any IANA actions. The benchmarking methodology defined herein:

* Does not define new protocol elements that require IANA registration
* Does not create new registries or modify existing ones
* Uses existing [MOQT] and [QUIC] protocol mechanisms without extension
* Defines message formats for benchmarking purposes only, which are not part of the [MOQT] protocol specification

The message type values defined in this document (0x01, 0x02, 0x03) are used solely within the context of benchmarking implementations and do not conflict with or extend the [MOQT] protocol message space.

7. Security Considerations

This document defines a benchmarking methodology and does not introduce new security vulnerabilities beyond those inherent in [MOQT] and [QUIC]. However, the following security considerations apply when implementing and running benchmarks.

8. References

8.1. 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-15, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-15>.

[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>.

8.2. Informative References

[QUIC] 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/rfc/rfc9000>.

Appendix A. Example Benchmark Results

Below are example results from running benchmark against existing relays.

A.1. Environment

Each benchmark test was using the same set of EC2 instances in AWS. The publisher and relay ran on a one EC2 instance while the clients ran on one or more separate EC2 instances.

* *EC2 Instance Type*: c8g.2xlarge (Graviton4 ARM processors (2.7/2.8GHz), 8 cores, 16GB ram

* *Networking*: UP to 15Gbps. All instances on same VPC subnet and no NAT gateway was used

A.2. Benchmark Tool

qperf (https://github.com/quicr/qperf) benchmark implementation was used.

A.3. Relays

Benchmark tests two open source relays using latest main branches on September 25th, 2025.

* LAPS (https://github.com/quicr/laps)

* Moxygen (https://github.com/facebookexperimental/moxygen)

A.4. Scenario 1: Single publisher audio

A.4.1. Config Profile

```
[Audio Datagram]
namespace           = perf/audio/{} ; MAY be the same across tracks,
                                    ; entries delimited by /
name                = 1             ; SHOULD be unique to other tracks
track_mode          = datagram      ; (datagram|stream)
priority            = 2             ; (0-255)
ttl                 = 5000          ; TTL in ms
time_interval       = 20            ; transmit interval in floating
                                    ; point ms
objects_per_group   = 1             ; number of objects per group >=1
first_object_size   = 120           ; size in bytes of the first object
                                    ; in a group
object_size         = 120           ; size in bytes of remaining objects
                                    ; in a group
start_delay         = 5000          ; start delay in ms - after control
                                    ; messages are sent and acknowledged
total_transmit_time = 35000         ; total transmit time in ms,
                                    ; including start delay
```

A.4.2. Results

Relay	Number of Subscribers	Notes
LAPS	2000	No dropped datagrams and CPU was under 95% on a single core
Moxygen	1000	No dropped datagrams and CPU was under 95% on a single core

Table 4: Scenario 1 Relay Results

A.5. Scenario 2: Single publisher audio/video

A.5.1. Config Profile

```
   [Audio Datagram]
   namespace          = perf/audio/{}  ; MAY be the same across tracks,
                                       ; entries delimited by /
   name               = 1              ; SHOULD be unique to other tracks
   track_mode         = datagram       ; (datagram|stream)
   priority           = 2              ; (0-255)
   ttl                = 5000           ; TTL in ms
   time_interval      = 20             ; transmit interval in floating
                                       ; point ms
   objects_per_group  = 1              ; number of objects per group >=1
   first_object_size  = 120            ; size in bytes of the first
                                       ; object in group
   object_size        = 120            ; size in bytes of remaining
                                       ; objects in a group
   start_delay        = 5000           ; start delay in ms - after control
                                       ; messages are sent and acknowledged
   total_transmit_time = 35000         ; total transmit time in ms

   [360p Video]
   namespace          = perf/video/{}  ; MAY be the same across tracks,
                                         ; entries delimited by /
   name               = 1            ; SHOULD be unique to other tracks
   track_mode         = stream       ; (datagram|stream)
   priority           = 3            ; (0-255)
   ttl                = 5000         ; TTL in ms
   time_interval      = 33.33        ; transmit interval in floating
                                     ; point ms
   objects_per_group  = 150          ; number of objects per group >=1
   first_object_size  = 21333        ; size in bytes of the first object
                                     ; in a group
   object_size        = 2666         ; size in bytes of remaining objects
                                     ; in a group
   start_delay        = 5000         ; start delay in ms - after control
                                     ; messages are sent and acknowledged
   total_transmit_time = 35000       ; total transmit time in ms
```

A.5.2. Results

Relay	Number of Subscribers	Notes
LAPS	1000	No dropped datagrams or stream data. CPU was under 95% on a single core
Moxygen	250	Datagrams started to get lost after 250 and subscriptions started to

			close and fail after 250. CPU on a single core was less than 85%

Table 5: Scenario 2 Relay Results

Acknowledgments

Thank you to Suhas Nandakumar for their reviews and suggestions.

Authors' Addresses

Tim Evens
Cisco
Email: tievens@cisco.com

Tomas Rigaux
Cisco
Email: trigaux@cisco.com

Scott Henning
Email: scotthenning00@gmail.com