Workgroup: Network Working Group
Internet-Draft:
draft-jennings-moq-quicr-arch-00
Published: 4 March 2022
Intended Status: Informational
Expires: 5 September 2022
Authors: C. Jennings   S. Nandakumar
cisco   Cisco

QuicR - Media Delivery Protocol over QUIC

Abstract

This specification outlines the design for a media delivery protocol over QUIC. It aims at supporting multiple application classes with varying latency requirements including ultra low latency applications such as interactive communication and gaming. It is based on a publish/subscribe metaphor where entities publish and subscribe to data that is sent through, and received from, relays in the cloud. The information subscribed to is named such that this forms an overlay information centric network. The relays allow for efficient large scale deployments.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Recently new usecases have emerged requiring higher scalability of delivery for interactive realtime applications and much lower latency for streaming applications and a combination thereof. On one side are usecases such as normal web conferences wanting to distribute out to millions of viewers and allow viewers to instantly move to being a presenter. On the other side are usescases such as streaming a soccer game to millions of people including people in the stadium watching the game live. Viewers watching an e-sports event want to be able to comment with mininal latency to ensure the interactivity aspects between what different viewers are seeing is preserved. All of these usescases push towards latencies that are in the order of 100ms over the natural latency the network causes.

Interactive realtime applications, such as web conferencing systems, require ultra low latency (< 150ms). Such applications create their own application specific delivery network over which latency requirements can be met. Realtime transport protocols such as RTP over UDP provide the basic elements needed for realtime communication, both contribution and distribution, while leaving aspects such as resiliency and congestion control to be provided by each application. On the other hand, media streaming applications are much more tolerant to latency and require highly scalable media distribution. Such applications leverage existing CDN networks, used for optimizing web delivery, to distribute media. Streaming protocols such as HLS and MPEG-DASH operates on top of HTTP and gets transport-level resiliency and congestion control provided by TCP.

This document outlines, QuicR, a unified architecture and protocol for data delivery that enables a wide range of realtime applications with different resiliency and latency needs without compromising the scalability and cost effectiveness associated with content delivery networks.

1.1. QuicR

The architecture defines and uses QuicR, a delivery protocol that is based on a publish/subscribe metaphor where client endpoints publish and subscribe to named objects that is sent to, and received from relays, that forms an overlay delivery network similar to what CDN provides today. The subscribe messages allow subscription to a name that includes a wildcard to match multiple published names, so a single subscribe can allow a client to receive publishes for a wide class of named objects. Objects are named such that it is unique for the relay/delivery network and scoped to an application.

QuicrR provides services based on application requirements (with the support of underlying transport, where necessary) such as estimation of available bandwidth, fragmentation and reassembly, resiliency, congestion control and prioritization of data delivery based on data lifetime and importance of data. It is designed to be NAT and firewall traversal friendly and can be fronted with load balancers.

The Relays are arranged in a logical tree (as shown below) where, for a given application, there is an origin Relay at root of the tree that controls the namespace. Publish messages are sent towards the root of the tree and down the path of any subscribers to that named object. QuicR is designed to make it easy to implement relays so that fail over could happen between relays with minimal impact to the clients and relays can redirect a client to a different relay.

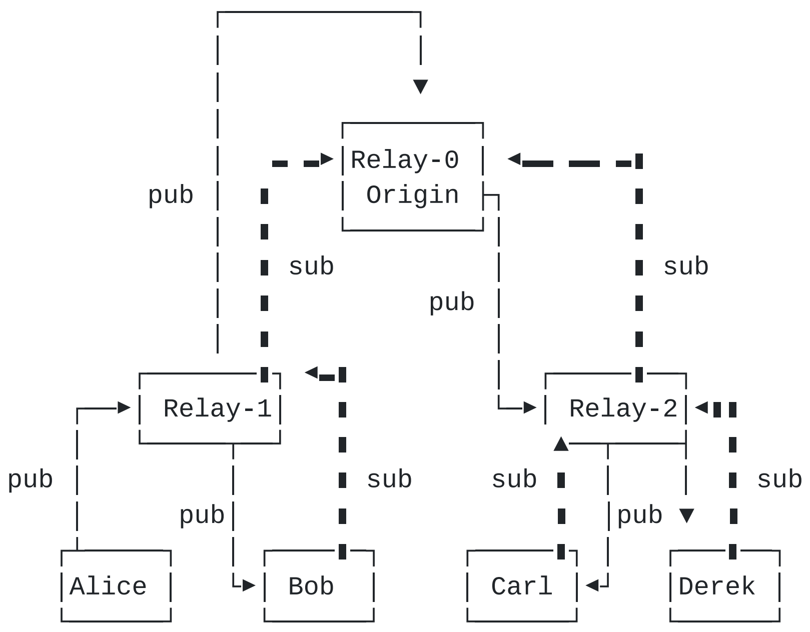

Figure 1: QuicR Delivery Tree

The design supports sending media and other named objects between a set of participants in a game or video call with under a hundred milliseconds of latency and meets the needs of conferencing systems. The design can also be used for large scale streaming to millions of participants with latency ranging from a few seconds to under a hundred milliseconds based on applications needs. It can also be used as low latency publish/subscribe system for real time systems such as messaging, gaming, and IoT.

2. Contributing

All significant discussion of development of this protocol is in the GitHub issue tracker at: https://github.com/fluffy/draft-jennings-moq-arch

QuicR is pronounced something close to “ (U+201C)quicker” (U+201D) but with more of a pirate "arrrr" at the end.

3. Terminology

* Relay Function: Functionality of the QuicR architecture, that implements store and forward behavior at the minimum. Such a function typically receives subscriptions and publishes data to the other endpoints that have subscribed to the named data. Such

functions may cache the data as well for optimizing the delivery experience.

* Relay: Server component (physical/logical) in the cloud that implements the Relay Function.

* Publisher: An endpoint that sends named objects to a Relay. [also referred to as producer of the named object]

* Subscriber: An endpoint that subscribes and receives the named objects. Relays can act as subscribers to other relays. Subscribers can also be referred to as consumers.

* Client/QuicR Client: An endpoint that acts as a Publisher, Subscriber, or both. May also implement a Relay Function in certain contexts.

* Named Object: Application level chunk of Data that has a unique Name, a limited lifetime, priority and is transported via QuicR protocol.

* Origin server: Component managing the QuicR namespace for a specific application and is responsible for establishing trust between clients and relays. Origin servers can implement other QuicR functions.

4. Advantages of QuicR

As its evident, QuicR and its architecture uses similar concepts and delivery mechanisms to those used by streaming standards such as HLS and MPEG-DASH. Specifically the use of a CDN-like delivery network, the use of named objects and the receiver-triggered media/data delivery. However there are fundamental characteristics that QuicR provides to enable ultra low latency delivery for interactive applications such as conferencing and gaming.

* To support low latency the granularity of the delivered objects, in terms of time duration, need to be quite small making it complicated for clients to request each object individually. QuicR uses a publish and subscription semantic along with a wildcard name to simplify and speed object delivery for low latency applications. For latency-tolerant applications, larger granularity of data, aka group of objects, can be individually requested and delivered without instantiating state in the backend.

* Some realtime applications operating in ultra low latency mode require objects delivered as and when they are available without having to wait for previous objects delayed due to network loss or out of order network delivery. QuicR supports Quic datagrams

based object delivery for this purpose. Note that QuicR also uses Quic stream for delivery of objects that are latency-tolerant.

* QuicR supports resiliency mechanisms that are more suitable for realtime delivery such as FEC and selective retransmission.

* QUIC's current congestion control algorithms need to be evaluated for efficacy in low latency interactive real-time contexts, specifically mechanisms such as slow start, multiplicative decrease and queue buildup drainage during BBR probing. Based on the results of the evaluation work, QuicR can select the congestion control algorithm suitable for the application's class.

* Published objects in QuicR have associated max-age that specifies the validity of such objects. max-age influences relay's drop decisions and can also be used by the underlying QUIC transport to cease retransmissions associated with the named object.

* Unlike streaming architectures where media contribution and media distribution are treated differently, QuicR can be used for both object contribution/publishing and distribution/subscribing as the split does not exist for interactive communications.

* QuicR supports "aggregation of subscriptions" to the named objects where the subscriptions are aggregated at the relay functions and allows "short-circuited" delivery of published objects when there is a match at a given relay function.

* QuicR allows publishers to associate a priority with objects. Priorities can help the delivery network and the subscribers to make decisions about resiliency, latency, drops etc. Priorities can used to set relative importance between different qualities for layered video encoding, for example.

* QuicR is designed so that objects are encrypted end-to-end and will pass transparently through the delivery network. Any information required by the delivery network, e.g priorities, will be included as part of the metadata that is accessible to the delivery network for further processing as appropriate.

5. QuicR architecture

A typical media delivery architecture based on QuicR enables delivery tree allowing :

* Publishing entities to publish named data

* Subscribers to express interest in the named objects

*Delivery tree made up of one or more Relays to allow the flow of the named objects.

In the following subsections, 2 common QuicR delivery tree architectures are non-normatively discussed

5.1. QuicR Delivery Network Architecture with Origin as the only Relay Function.

```
                       +-------------+
                       |Relay        |
    +----------------> |Origin:tw.com|-----+
    |                  +-------------+     |
    |                                ^     |
    |pub:tw.com/ch22/3/1             |     |
    |                                |     |
    |               sub:tw.com/ch22/*|     |
    |                                |     |pub:tw.com/ch22/3/1
    |                                |     v
+-----------+                      +----------+
| Publisher |                      |Subscriber|
+-----------+                      +----------+
```

The above picture shows QuicR delivery network for an hypothetical streaming architecture rooted at the Origin Relay (for the domain tw.com). In this architecture, the media contribution is done by publishing named objects corresponding to channel-22 to the ingest server at the Origin Relay. Media consumption happens via subscribes sent to the Origin Relay to the wildcarded name (ch22/*) for all media streams happening over the named channel-22. The media published either by the source publisher or the Relay (as Publisher) might be encoded into multiple qualities.

5.2. QuicR Delivery Network Architecture

```
                  +--------+
           +----> |Realay-O| <----------------+
           |      +--------+                  |
           |       ^       |                  |sub:alice-low
 pub:alice-hi      |       pub:alice-hi       |sub:alice-hi
 pub:alice-low     |       pub:alice-low      |
           | sub:alice-low |                  |
           |       |       |                  |
          +---------+      |  +------------------------+
  +------>| Relay-A |      +->|    Relay-B             |
  |       +---------+         +------------------------+
  |           |  ^              |    ^          |    ^
 pub1:alice-hi|  |              |    |          |    |
 pub2:alice-low  |              |    |          |    |
  |           |  |              |    |          |    |
  |          pub:alice-low     pub:alice-hi,low pub:alice-hi,low
  |           |  |              |    |          |    |
  |           | sub:alice-low   |   sub:alice*  |   sub:alice*
  |           v  |              v    |          v    |
  +------+    +---+              +----+         +-----+
  | Alice|    |Bob|              |Carl|         |Derek|
  +------+    +---+              +----+         +-----+
```

The above picture shows QuicR media delivery tree formed with multiple relays in the network. The example has 4 participants with Alice being the publisher and rest being the subscribers. Alice's is capable of publishing video streams at 2 qualities identified by their appropriate names. Bob subscribes to a low resolution video feed from alice, where as Carl/Derek carryout wildcard subscribes to all the qualities of video feed published by Alice. All the subscribes are sent to the Origin Relay and are saved at the on-path Relays, this allowing for "short-circuited" delivery of published data at the relays. In the above example, Bob gets Alice's published data directly from Relay-A instead of hairpinning from the Origin Relay. Carl and Derek, however get their video stream relayed from Alice via Origin Relay and Relay-B.

6. Names and Named Objects

Names are basic elements with in the QuicR architecture and they uniquely identify objects. Named objects can be cached in relays in a way CDNs cache resources and thus can obtain similar benefits such caching mechanisms would offer.

6.1. Objects Groups

Objects with in QuicR belong to a group. A group (a.k.a group of objects) represent an independent composition of set of objects, where there exists dependency relationship between the objects within the group. Groups, thus can be independently consumable by the subscriber applications.

A typical example would be a group of pictures/video frames or group of audio samples that represent synchronization point in the video conferencing example.

Latency-tolerant applications can request individual group of objects allowing delivery of objects without instantiation of persistent state within the delivery network. This is important for the preservation of the scalability of delivery networks at levels similar to what is currently available when streaming protocols such as HLS/HTTP are used.

6.2. Named Objects

The names of each object in QuicR is composed of the following components:

1. Domain Component

2. Application Component

3. Group ID Component

4. Object ID Component

Domain component uniquely identifies a given application domain. This is like a HTTP Origin and uniquely identifies the application and a root relay function. This is a DNS domain name or IP address combined with a UDP port number mapped to into the domain. Example: sfu.webex.com:5004.

Application component is scoped under a given Domain. This component identifies aspects specific to a given application instance hosted under a given domain (e.g.meeting identifier, which movie or channel, media type or media quality identifier).

Inside each Application Component, there is a set of groups. Each group is identified by a monotonically increasing integer. Inside of each Group, each object is identified by another monotonically increasing integer inside that group. The groupID and objectID start at 0 and are limited to 16 bits long.

Example: In the example below, the domain component identifies acme.meeting.com domain, the application component identifies an instance of a meeting under this domain, say "meeting123", and high resolution camera stream from the user "alice". It also identifies the object 17 under group 15.

```
quicr://acme.meeting.com/meetingl23/alice/cam5/HiRes/15/17
```

6.3. Name Hashes

All Names need to hash or map down to 128 bits. This allows for:

* compact representation for efficient transmission and storage,

* cache friendly datatypes (like Keys in CDN caches) for storage and lookup purposes and,

* enable rapid data lookup at the relays based on partial as well as whole names (wildcard support).

```
┌─────────────┬───────────────┬───────────────┬─────────────┐
│ Domain      │ Application   │ GroupID       │ ObjectID    │
│ Component   │ Component     │ Component     │ Component   │
└─────────────┴───────────────┴───────────────┴─────────────┘

    48 bits        48 bits         16 bits       16 bits
```

Figure 2: QuicR Name

This is done by hashing the origin to first 48 bits. Any relay that forms an connection to an new origin needs to ensure this does not collide with an existing origin. The application component is mapped to the next 48 bits and it is the responsibility of the application to ensure there are no collisions within a given origin. Finally the group ID and object ID each map to 16 bits.

Design Note: It is possible to let each application define the size of these boundaries as well as sub boundaries inside the application component but for sake of simplicity it is described as fixed boundaries for now.

Wildcard search simply turns into a bitmask at the appropriate bit location of the hashed name.

The hash names are key part of the design for allowing small objects without adding lots of overhead and for efficient implementation of Relays.

6.4. Wildcarding with Names

QuicR allows subscribers to request for media based on wildcard'ed names. Wildcarding enables subscribes to be made as aggregates instead of at the object level granularity. Wildcard names are formed by skipping the right most segments of names.

For example, in an web conferencing use case, the client may subscribe to just the origin and meetingId to get all the media for a particular conference as indicated by the example below. The example matches all the named objects published as part of meeting123.

quicr://acme.meeting.com/meeting123/*

When subscribing, there is an option to tell the relay to one of:

A. Deliver any new objects it receives that matches the name

B. Deliver any new objects it receives and in addition send any previous objects it has received that are in the same group that matches the name.

C. Wait until an object that has a objectID that matches the name is received then start sending any objects that match the name.

7. Objects

Once a named object is created, the content inside the named object can never be changed. Objects have an expiry time after which they should be discarded by caches. Objects have an priority that the relays and clients can use to sequence the sending order. The data inside an object is end-to-end encrypted whose keys are not available to Relay(s).

8. Relays

The Relays receive subscriptions and intent to publish request and forward them towards the origin. This may send the messages directly to the Origin Relay or possibly traverse another Relay. Replies to theses message follow the reverse direction of the request and when the Origin gives the OK to a subscription or intent to publish, the Relay allows the subscription or future publishes to the Names in the request.

Subscription received are aggregated. When a relay receives a publish request with data, it will forward it both towards the Origin and to any clients or relays that have a matching subscription. This "short circuit" of distribution by a relay before the data has even reached the Origin servers provides significant latency reduction for nearby client.

The Relay keeps an outgoing queue of objects to be sent to the each subscriber and objects are sent in priority order.

Relays MAY cache some of the information for short period of time and the time cached may depend on the Origin.

9. QuicR Usage Design Patterns

This section explains design patters that can be use to build applications on top of QuicR.

9.1. QuicR Manifest Objects

Names can optionally be discovered via manifests. In such cases, the role of the manifest is to identify the names as well as aspects pertaining to the associated data in a given usage context of the application.

* Typically a manifest identifies the domain and application aspects for the set of names that can be published.

* The content of Manifest is application defined and end-to-end encrypted.

* The manifest is owned by the application's origin server and are accessed as a protected resources by the authorized QuicR clients.

* The QuicR protocol treats Manifests as a named object, thus allowing for clients to subscribe for the purposes of bootstrapping into the session as well as to follow manifest changes during a session [new members joining a conference for example].

* The manifest has well known name on the Origin server.

Also to note, a given application might provide non QuicR mechanisms to retrieve the manifest.

9.2. QuicR Video Objects

Most video applications would use the application component to identity the video stream, as well as the encoding point such as resolution and bitrate. Each independently decodable set of frames would go in a single group, and each frame inside that group would go in a separate named object inside the group. This allows an application to receive a given encoding of the video by subscribing just to the application component of the Name with a wildcard for group and object IDs.

This also allows a subscription to get all the frames in the current group if it joins lates, or wait until the next group before starting to get data, based on the subscription options. Changing to a different bitrate or resolution would use a new subscription to the appropriate Name.

The QUIC transport that QuicR is running on provides the congestion control but the application can see what objects are received and determine if it should change it's subscription to a different bitrate application component.

Todays video is often encoded with I-frames at a fixed internal but this can result in pulsing video quality. Future system may want to insert I-frames at each change of scene resulting in groups with a variable number of frames. QuicR easily supports that.

9.2.1. RUSH over QuicR

RUSH is an application-level protocol for ingesting live video. This section defines at a higher level how aspects of the RUSH protocol could be realized with QuicR.

RUSH's video frame is equivalent to QuicR video object that represents an instance of encoder output. For video ingestion, the RUSH publisher can assign the same groupID for all the frames generated between the I-Frame boundaries and the RUSH's frameID can be directly mapped to QuicR's object ID. RUSH multistream mode can enabled by publishing each frame over QUIC Stream indicated via QuicR API, since QuicR supports both the QUIC Datagram and QUIC Stream modes of transport.

The identifiers for the track and session forms the application component of the name.

Below is an example that shows RUSH's video frame mapped to QuicR name for the session1, track 12 and video-id that maps to a given encoding. The groupID and objectID follow the encoder output. The payload of the published message will be formed by the actual encoded data along with metadata such as PresentationTimeStamp (PTS) and so on.

```
quicr://rush-ingest-server/session1/track12/video-id/group1/object10
```

9.2.2. Warp over QuicR

Warp is a segmented live video transport protocol. Warp maps live media to QUIC streams based on the underlying media encoding.

Conceptually, each Warp video media segment maps to QuicR groupID and frames within segment to QuicR objectID. Warp video media segments are made up of I-Frames and zero or more related frames, which corresponds to QuicR group of objects. QuicR named objects correspond to these frames mapped to these segments and are published individually. For a given channel and video quality, a segment and its frames can be mapped to QuicR name as below:

```
quicr://twitch.com/channel-fluffy/video-quality-id/group12/object0
```

In this example, groupID 12 maps to Warp segmentId 12 and objectID 0 corresponds to I-frame within that segment.

9.3. QuicR Audio Objects

Each small chuck of audio, such as 10 ms, can be its own QuicR object.

Future sub 2 kbps audio codecs may take advantage of a rapidly updated model that are needed to decode the audio which could result in audio needing to use groups like video to ensure all the objects needed to decode some audio are in the same group.

9.4. QuicR Game Moves Objects

Some games send out a base set of state information then incremental deltas to this. Each time a new base set is sent, a new group can be formed and each increment change as an Object in the group. When new players join, they can subscribe to sync up to the latest state by subscribing to the current group and the incremental changes that follow.

9.5. Messaging Objects

Chat applications and messaging system can form a manifest representing the roster of the people in a given channel or talk room. The manifest can provide information on the application component of the Quicr Name for user that are contributing messages. A subscription to each application such component enables reception of each new message. Each message would be a single object. Typically QuicR would be used to get the recent messages and then a more traditional HTTP CDN approach could be used to retrieve copies of all the older objects.

10. Security Considerations

The links between Relay and other Relays or Clients can be encrypted, however, this does not protect the content from Relays. To mitigate this all the objects needs to be end-to-end encrypted with a keying mechanism outside the scope of this protocol. For may applications, simply getting the keys over HTTPS for a particular object/group from the origin server will be adequate. For other applicants keying based on MLS may be more appropriate. Many applications can leverage the existing key managed schemes used in HLS and DASH for DRM protected content.

Relays reachable on the Internet are assumed to have a burstiness relationship with the Origin and the protocol provides a way to verify that any data moved is on behalf of a give Origin.

Relays in a local network may choose to process content for any Origin but since only local users can access them, their is a way to mange which applications use them.

Subscriptions need to be refreshed at least every 5 seconds to ensure liveness and consent for the client to continue receiving data.

11. Protocol Design Considerations

11.1. HTTP/3

It is tempting to base this on HTTP but there are a few high level architectural mismatches. HTTP is largely designed for a stateless server in a client server architecture. The whole concept of the PUB/SUB is that the relays are *not* stateless and keep the subscription information and this is what allows for low latency and high throughput on the relays.

In todays CDN, the CDN nodes end up faking the credentials of the origin server and this limits how and where they can be a deployed. A design with explicitly designed relays that do not need to do this, while still assuming an end to end encrypted model so the relays did not have access to the content makes for a better design.

It would be possible to start with something that looked like HTTP as the protocol between the relays with special conventions for wildcards in URLs of a GET and ways to stream non final responses for any responses perhaps using something like multipart MIME. However, most of the existing code and logic for HTTP would not really be usable with the low latency streaming of data. It is probably much simpler and more scalable to simply define a PUB/SUB protocol directly on top of QUIC.

11.2. QUIC Streams and Datagrams

There are pro and cons to mapping object transport on top of streams or on top of QUIC datagrams. The working group would need to sort this out and consider the possibility of using both for different types of data and if there should be support for a semi-reliable transport of data. Some objects, for example the manifest would always want to be received in a reliable way while other objects may have to be realtime.

11.3. QUIC Congestion Control

The basic idea in BBR of speeding up to probe then slowing down to drain the queue build up caused during probe can work fine with real time applications. However the the current implementations in QUIC do not seem optimized for real time applications and have some times where the slow down causes too much jitter. To not have playout drops, the jitter buffers adds latency to compensate for this. Probing for the RTT has been one of the phases that causes particular problems for this. To reduce the latency of QUIC, this work should coordinate with the QUIC working group to have the QUIC working group develop congestion control optimizations for low latency use of QUIC.

11.4. Why not RTP

RTP has several desirable properties that optimize the transport of media over networks, including media payload formats explicitly designed for network packets, transport feedback, packet loss resilience mechanisms, multiplexing, and strong security. It also has experimental congestion control (CC) algorithms explicitly designed for media delivery (RMCAT), without the issues described above in BBR.

However, these properties have less value in the context of QuicR for the following reasons. QUIC adequately handles multiplexing, security, and transport feedback (except ack timestamps which require extensions proposed in drafts that have not yet been adopted by the QUIC WG). QUIC lacks CC and resilience mechanisms optimized for media, but direct reuse of unaltered RTP mechanisms is not practical, so these aspects must be redesigned in the context of QUIC anyway, although they can leverage learnings from RTP.

Finally, and most significantly, RTP media payload formats that were optimized for network packets are less useful in QuicR since a primary goal is to unify the streaming and real-time media delivery protocols. Streaming protocols use "container" formats like CMAF, ISOBMFF, etc. Codecs always first define their core "elementary" bitstream format, then define their container format binding, and finally define their RTP payload format binding. These always differ. The differences are not significant enough to justify supporting both, so QuicR only supports the container format binding.

It is also interesting to observe that the use of RTP inadvertently leads to media description and negotiation using SDP. Such complexity is justifiable when huge variation exists between clients' capabilities with very basic common lowest denominators. Today, and while variations still exist, streamlining media capabilities into reasonable capability sets that are declared by publishers and subscribed to by subscribers is very feasible and is how the streaming applications do operate. Simpler forms can and should be used for media declarations. As a very wise guru once put it "RTP is an gateway drug to SDP and friends done't let friends try to debug SDP".

In summary, the desirable aspects of RTP are absorbed into QUIC or QuicR layers rather than direct encapsulation of RTP.

Appendix A. Acknowledgments

Thanks to Nermeen Ismail, Mo Zanaty for contributions and suggestions to this specification.

Authors' Addresses

Cullen Jennings
cisco
Canada

Email: fluffy@iii.ca

Suhas Nandakumar
Cisco

Email: snandaku@cisco.com