QUIC
Internet-Draft
Intended status: Standards Track
Expires: 13 April 2026

袁靖昊 (J. Yuan)
肖磊 (L. Xiao)
Bytedance
M. Bishop
Akamai Technologies

10 October 2025

Exchanging Congestion Control Data in QUIC
draft-yuan-quic-congestion-data-00

Abstract

This draft defines a new transport frame which enables consenting endpoints to share congestion control state about the network connection for various purposes. It also allows an endpoint's own congestion control state to be echoed back to it by a peer for consideration at the beginning of a future connection.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-yuan-quic-congestion-data/.

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

Source for this draft and an issue tracker can be found at https://github.com/MikeBishop/quic-samsara.

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 13 April 2026.

Table of Contents

1. Introduction

All Internet transports are required to either use a congestion control algorithm, or to constrain their rate of transmission [RFC8085]. Most congestion control algorithms take time to ramp up the sending rate, called the "Slow-Start phase." This defines a time in which a sender intentionally uses less capacity than might be available so as to avoid or limit overshoot of the available capacity for the path. This is because any overshoot can have detrimental effects both for the current flow and for any other flows with which it shares network bottlenecks.

At the same time, using less capacity than is available necessarily limits performance early in the connection. Careful Resume [CAREFUL-RESUME] is a mechanism whereby remembered congestion control parameters can be validated as potentially applicable to a new connection, probed, and ultimately used to grow the congestion window more rapidly than slow-start would otherwise permit.

One optimization approach is to use historical congestion information to provide the congestion algorithm with reliable input to help it exit the slow start phase. The most direct and reliable information can be samples collected during the congestion algorithm, such as the congestion window size and probe bandwidth.

While clients often connect to a manageable number of servers and can retain such state, servers typically service orders of magnitude more clients and cannot feasibly retain such information. Further, servers are often deployed with many instances and attempting to coordinate the sharing of this information between them may prove impractical. Thus, for a server to implement Careful Resume, some external means of recalling its previous state is useful.

This document specifies a mechanism which allows a QUIC [QUIC] endpoint to periodically export its congestion control state, optionally in an integrity-protected manner. This exported state is sent to the peer in a CONGESTION_DATA frame. When establishing a subsequent connection, an endpoint with persistent storage can

include this data in a CONGESTION_DATA_RECALL frame in its 0-RTT or 1-RTT packets, assisting its peer to recall the information necessary to perform Careful Resume.

This mechanism is comparable to HTTP cookies, [COOKIES], but for transport information. This data may also be useful for application-layer purposes, such as adaptive-bit-rate video. The exported information is readable by the peer and can be exposed to the application through local interfaces.

1.1. Peer Visibility

The peer’s viewpoint of a connection can be useful for debugging and as additional information to be considered by on-path entities such as congestion controllers and application-layer protocols. Therefore, this extension deliberately does not encrypt the data reported to the peer. Instead, the data is provided in cleartext with an optional integrity tag.

If a server wishes to recall information about past connections without sharing that data with the client, this information can already be encoded in address validation tokens without requiring the cooperation of the client; see Section 8.1.3 of [QUIC].

1.2. 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.

This document also uses terminology defined in [QUIC] and [QUIC-TLS], in particular the frame layout notation from Section 1.3 of [QUIC].

2. Transport Parameter

Desire and willingness to receive the frames defined in this specification is indicated by means of the following QUIC transport parameter:

support_congestion_data(i)

The support_congestion_data value is a variable-length integer that encodes these three one-bit flags:

CONSUME (0x01): This indicates that the sender is interested in

receiving CONGESTION_DATA frames for its own uses during the current connection, independent of the receiver's ability to reuse the data in the future.

CACHE (0x02): This indicates that the sender is willing to receive CONGESTION_DATA frames and potentially return the contents in a CONGESTION_DATA_RECALL frame on a subsequent connection.

CONSIDER (0x04): This indicates that the sender is willing to have values it may have provided on a previous connection returned to it in a CONGESTION_DATA_RECALL frame.

All other bits MUST be set to zero when sending and MUST be ignored on receipt.

The default for this parameter is 0, which indicates that the endpoint does not support CONGESTION_DATA or CONGESTION_DATA_RECALL frames. A value other than 0 indicates that the endpoint supports the indicated frame types and is willing to receive such frames on this connection.

An endpoint MUST NOT send CONGESTION_DATA or CONGESTION_DATA_RECALL frames until it has received the support_congestion_data transport parameter with a non-zero value during the handshake (or during a previous handshake if 0-RTT is used).

An endpoint MUST NOT send CONGESTION_DATA frames to a peer which did not set the CONSUME or CACHE flags. An endpoint MUST NOT send CONGESTION_DATA_RECALL frames to a peer which did not set the CONSIDER flag. An endpoint that receives a frame for which it has not indicated support via the transport parameter MUST terminate the connection with an error of type PROTOCOL_VIOLATION.

3. Network Statistics

Each network statistic is structured as a TLV:

```
Network Statistic structure {
  Type (i),
  Length (i),
  Value (..),
}
```

This structure includes the following fields:

Type: Indicates the statistic being offered, encoded as a variable-length integer.

Length: The length of the Value field in bytes, encoded as a variable-length integer.

Value: A Type-specific value carrying the payload of the indicated statistic.

This specification defines a number of initial statistics, but additional statistics can be added by registering a value in the appropriate registry (see Section 6). An implementation MUST ignore any statistics it cannot understand, but MAY decline to return protected statistics to a peer if it cannot verify that it is willing to share the contained information.

A receiver may get a message with multiple occurrences of a particular TLV value. If the values are identical, the reciever SHOULD ignore them. If they differ, and one of the values is protected by an integrity tag, the receiver SHOULD treat this as an attack and close the connection. If none of the instances are integrity-protected, the receiver MAY ignore them, use only one of the instances, or close the connection as it determines to be most appropriate.

In the sub-section below, only the names are used; the numeric value that appears in the protocol is defined in Section 6.3.

3.1. Timestamp

The Timestamp statistic indicates the time at which the sender generated this frame. This can be used on future connections to determine whether the recalled statistics are recent enough to be useful.

Note Format of the timestamp is TBD.

3.2. Path Tuple

The Path Tuple statistic encodes an identifier of the path on which these statistics were generated. Knowing the connection addresses as seen from the peer’s perspective can be useful for a number of scenarios (e.g., [STUN]). The reciever MAY use this to to compare similarity of the previous endpoints to those of a new connection will when deciding if returned statistics might be applicable to a new connection.

The structure of the value is:

```
Path Endpoint structure {
  Type (i) = 0x202,
  Length (i) = 12/36,
  Local IP (4/16),
  Local Port (2),
  Remote IP (4/16),
  Remote Port (2),
}
```

NOTE I am confused about this structure. Is the Type and Length from the TLV? If so, should "0x202" be the assigned number "0xca" ?

It contains the following fields:

Local IP: The sender’s IP address on the associated transport path, encoded as either 4 or 16 bytes depending on IP version.

Local Port: The sender’s port number on the associate transport path, encoded as a two-byte integer.

Remote IP: The receiver’s IP address as observed by the sender on the associated transport path, encoded as either 4 or 16 bytes depending on IP version.

Remote Port: The receiver’s port number as observed by the sender on the associated transport path

The IP version being used can be inferred from the length of the payload.

3.3. Slow Start Status

The Slow Start Status statistic indicates whether the sender’s congestion controller is in the Slow Start phase. The value is a single byte, set to 0x00 if the sender is not in Slow Start and 0x01 if the sender is in Slow Start.

3.4. Network Type

The Network Type statistic indicates the sender’s understanding of its network access medium, encoded as a single byte value. Note that this is purely advisory, since applications will only be aware of the local network at best.

The defined values are at Section 6.2.

3.5. Maximum Congestion Window

The Maximum Congestion Window statistic indicates the maximum congestion window (CWD) sampled within the observation period, measured in bytes. It is encoded as a variable-lenth integer. CWD is a key metric in congestion control algorithms, as it represents the amount of unacknowledged data that a sender can have in flight on the network. A larger CWD generally allows for a higher sending rate.

3.6. Maximum In-Flight Data

The Maximum In-Flight Data statistic indicates the maximum in-flight data sampled within the observation period, measured in bytes. It is encoded as a variable-length integer. This represents the highest number of bytes sent by the sender that have not yet been acknowledged by the receiver during the measurement period. This metric provides insight into the actual amount of data in transit at any given time, which can be useful for diagnosing network performance issues.

3.7. Smoothed RTT

The Smoothed RTT statistic indicates the most recent smoothed Round-Trip Time (RTT) within the observation period, measured in milliseconds. It is encoded as a variable-length integer. It is calculated as defined in [RFC9002]. RTT is a key metric for congestion control, estimating the time it takes for a packet to travel from the sender to the receiver and back. The smoothed RTT calculation accounts for both the latest RTT and a historical average, helping to dampen the effect of short-term network fluctuations.

3.8. Minimum RTT

The Minimum RTT statistic indicates the minimum RTT sampled within the observation period, measured in milliseconds. It is encoded as a variable-length integer, This metric provides a baseline for the best-case network latency observed during the measurement period. A low minimum RTT can indicate a stable and efficient network path, while a high one might suggest persistent latency issues.

3.9. RTT Variance

The RTT Variance statistic indicates the most recent RTT deviation within the observation period, measured in milliseconds. It is encoded as a variable-length integer. It is calculated as defined in [RFC9002]. This metric quantifies the variability of the RTT, providing insight into network jitter and stability. A low RTT variance suggests a consistent network path, while a high value indicates significant fluctuations in network latency.

3.10. Latest Bandwidth

The Latest Bandwidth statistic indicates the current raw throughput of the connection, measured in kilobits per second (kbps). It is encoded as a variable-length integer. This metric represents the instantaneous sending capacity as perceived by the sender and is a crucial input for congestion control algorithms.

3.11. Maximum Bandwidth

The Maximum Bandwidth statistic indicates the maximum raw throughput sampled within the observation period, measured in kbps. It is encoded as a variable-length integer. This metric provides a view of the peak network capacity observed during the measurement period, which can be useful for understanding the best possible performance on the current network path.

3.12. Throughput

The Throughput statistic indicates the useful throughput for data, excluding retransmissions, within the observation period, measured in kbps. It is encoded as a variable-length integer. This metric is a measure of the effective data rate delivered to the receiver’s application layer. It isolates the useful data rate, providing a more accurate measure of application-level performance than the raw sending rate, which includes retransmitted data.

3.13. Send Rate

The Send Rate statistic indicates the sending rate for all data, including retransmissions, within the observation period, measured in kbps. It is encoded as a variable-length integer. This metric provides a measure of the total data rate at which the sender is transmitting data. It is useful for understanding the sender’s total load on the network.

3.14. Receive Rate

The Receive Rate statistic indicates the receiving rate within the observation period in kbps. It is encoded as a variable-length integer. This metric measures the total rate at which the receiver is acknowledging data, including both new data and retransmissions. It is useful for understanding the receiver's perspective on the data flow and can be used to compare against the sender's rate to identify potential bottlenecks.

3.15. Input Rate

The Input Rate statistic indicates the input bitrate from the application layer within the observation period in kbps. It is encoded as a variable-length integer. This metric represents the rate at which data is being provided to the receiving application. It gives an end-to-end view of the application data flow.

3.16. Loss Rate

The Loss Rate statistic indicates the arithmetic mean of the packet loss rate samples within the observation period. The value is expressed as a percentage at 0.1% resolution within a range of 0 to 1000 inclusive. It is encoded as a variable-length integer. This metric provides a clear measure of the quality of the network path, as it quantifies the proportion of packets that are sent but not received. A high loss rate often indicates network congestion or instability.

3.17. Buffer Length

The Buffer Length statistic indicates the current amount of data cached by the QUIC connection layer buffer when the observation frame is generated in bytes. It is encoded as a variable-length integer. This metric reflects the amount of data the sender is holding in its buffer before transmission, which can be an important indicator of the sender's ability to keep up with the application's sending rate and can also be a sign of network congestion.

4. Frame Types

4.1. CONGESTION_DATA Frames

CONGESTION_DATA frames (type TBD1) provide a list of Network Statistics values which the sender chooses to share about the state of the network connection from its viewpoint.

```
CONGESTION_DATA Frame {
  Type (i) = TBD1,
  Protected Count (i),
  Protected Network Statistics (..) ...,
  [Integrity Tag (1..)],
  Unprotected Count (i),
  Unprotected Network Statistics (..) ...,
}
```

CONGESTION_DATA frames contain the following fields:

Protected Count: A variable-length integer representing the number of Network Statistics in the Protected Network Statistics field.

Protected Network Statistics: A sequence of Network Statistics objects whose length is given by the Protected Count.

Integrity Tag: A message integrity check, as described in Section 4.1.2. This field is absent if Protected Count is zero. While this document provides some examples, the format of the check MUST be treated as opaque by the receiver.

Unprotected Count: A variable-length integer representing the number of Network Statistics in the Unprotected Network Statistics field.

Unprotected Network Statistics: A sequence of Network Statistics objects whose length is given by Unprotected Count.

CONGESTION_DATA frames are not retransmittable, though a loss event might trigger the generation of a new CONGESTION_DATA frame; see Section 4.1.1.

CONGESTION_DATA frames can be sent at any point in the connection after 0-RTT or 1-RTT keys have been established, though useful data will likely not be available until at least one round-trip has occurred. If a CONGESTION_DATA frame is received in an Initial or Handshake packet, it MUST be treated as a connection error of type PROTOCOL_VIOLATION.

4.1.1. Sending Network Statistics

If an endpoint wishes to receive CONGESTION_DATA_RECALL frames on future connections with the peer and the peer has set the CACHE flag, the endpoint MAY send CONGESTION_DATA frames containing the values it wishes to recall in future connections in the Protected Network Statistics field. It MAY send additional CONGESTION_DATA frames when these values have changed significantly and it wishes to update the stored values, or when a previous CONGESTION_DATA frame is declared

lost.

If the peer has set the CONSUME flag, an endpoint SHOULD send CONGESTION_DATA frames periodically throughout the connection's lifetime. However, an implementation SHOULD NOT send additional CONGESTION_DATA frames if the connection has been idle since the last such frame was sent.

In addition to any values the endpoint placed in Protected Network Statistics, the endpoint includes such other values as it is willing to provide in the Unprotected Network Statistics field.

If an endpoint sends multiple CONGESTION_DATA frames, it is not required to include the same set of network statistics in each frame. For example, some statistics are more useful sent at a regular frequency, while others only need to be sent if they have changed significantly from the last value known to have been received. However, as the server does not control which CONGESTION_DATA will be cached, it SHOULD include the same Protected Network Statistics fields in each frame.

4.1.2. Integrity Tag

The integrity tag is calculated over the Protected Count and Protected Network Statistics field by the sender. This field is a variable-length set of bytes, whose format is known only to the sender. The purpose of this field is to provide suitable assurance to the sender that, when the statistics are later sent back to it through the CONGESTION_DATA_RECALL frame, that they have not been modified. This is often called a "message authentication code" (MAC). To emphasize that only a portion of a message is protected, this document does not use that term.

The algorithm for generating and verifying an integrity tag MAY depend on the ordering of the Protected fields although some implementations may perform a simple canonicalization by sorting the statistics by type identifier. Because of this, receivers SHOULD NOT modify the content or ordering of any of the Protected statistics in any way, unless they have out-of-band knowledge that it is safe to do so.

If the server has a nonce or other private material, it can hash that with the incoming Protected fields and use that as the outgoing Integrity tag. This can be either a simple hash of both parts, or the HMAC keyed hash [RFC2104] can be used.

Being able to change algorithms without large-scale protocol modifications is important. Servers may wish to use a fixed-number of leading bytes to indicate the algorithm they are using. It is also a best practice to generate new private data periodically, while still allowing old messages to be validated. To handle this, it is a good idea to use a fixed number of secondary bytes to act as a key or nonce identifier.

A sample implementation is provided in Appendix A.

4.2. CONGESTION_DATA_RECALL Frames

CONGESTION_DATA_RECALL frames (type TBD2) contain a list of Network Statistics values which the sender received from the recipient during a previous connection.

This frame SHOULD be sent as early as possible in the connection once 0-RTT or 1-RTT keys are available. While the frame MAY be sent at any point in the connection, if it arrives after the recipient has exited slow-start the values it contains will likely not be useful.

```
CONGESTION_DATA_RECALL Frame {
  Type (i) = TBD2,
  Protected Count (i),
  Protected Network Statistics (..) ...,
  Integrity Tag (1..),
}
```

NOTE Do we want to allow unprotected statistics here also, with the caveat that the receiver may reject them, or even the whole message?

CONGESTION_DATA_RECALL frames contain the following fields:

Protected Count: A variable-length integer representing the number of Network Statistics in the Protected Network Statistics field, received in a CONGESTION_DATA frame from the peer on a previous connection.

Protected Network Statistics: A sequence of Network Statistics objects whose length is given by Protected Count, received in a CONGESTION_DATA frame from the peer on a previous connection.

Integrity Tag: The integrity tag, received in a CONGESTION_DATA frame from the peer on a previous connection. See Section 4.1.2.

If a CONGESTION_DATA_RECALL frame is received in an Initial or Handshake packet, it MUST be treated as a connection error of type PROTOCOL_VIOLATION.

4.2.1. Recalling Network Statistics

Upon receipt of a CONGESTION_DATA_RECALL frame, an endpoint computes the expected Integrity Tag value as in Section 4.1.2. If the Integrity Tag value does not match, the frame is ignored.

If the tag is acceptable, the endpoint takes the network statistics contained in the frame and incorporates them into its congestion control strategy. For example, it might exit the Reconnaissance Phase of Careful Resume [CAREFUL-RESUME]. The specifics of how this is done are outside the scope of this extension.

Endpoints MUST NOT process more than one CONGESTION_DATA_RECALL frame on a connection. Subsequent CONGESTION_DATA_RECALL frames MUST be ignored without processing, regardless of whether the first frame was valid.

5. Security Considerations

Clients choosing to return network statistics to a server provide a potential tracking mechanism. However, this tracking mechanism provides no additional capabilities to a server beyond those already enabled by the address validation tokens defined in Section 8.1.3 of [QUIC]. While address validation tokens are opaque and can contain any data the server might wish to recall, the statistics being transported by this mechanism are visible to the clients. Clients can inspect the values to ensure that nothing objectionable is being saved; implementations MAY choose not to send CONGESTION_DATA_RECALL packets which contain statistics they cannot interpret.

Clients SHOULD NOT send CONGESTION_DATA_RECALL packets on connections where they would not have sent an Address Validation token if one were available. A client MAY also decide not to send the packet if the length of the integrity tag does not correspond to a digest length and a few additional bytes. This is admittedly inelegant. and could be avoided if the format of the tag were publicly defined, and an IANA registry for tag algorithms defined.

Clients SHOULD discard stored network statistics when other potential tracking mechanisms (e.g. HTTP Cookies) are cleared by the user.

6. IANA Considerations

IANA is requested to take the following actions, replacing "ThisRFC" with the RFC number when assigned.

6.1. New QUIC Transport Parameters Entry

Add a new entry to the "QUIC Transport Parameters" registry with the following values:

Field Name	Value
Value	TBD
Parameter Name	support_congestion_data
Status	permanent
Specification	ThisRFC
Date	TBD
Change Controller	IETF
Contact	quic@ietf.org
Notes	None

Table 1

6.2. New QUIC Network Type Registry

A new registry "QUIC Network Type" is created with the following fields:

Value: Numeric value

Meaning: Brief textual description

Reference: A pointer to the defining document

The registration policy for this registry is "Specification Required" as described in [RFC8126], Section 4.6.

The initial value of the registry is as follows:

Value	Meaning	Reference
0x00	Reserved	ThisRFC
0x01	Wire/Ethernet	ThisRFC
0x02	Reserved	ThisRFC
0x03	WLAN	ThisRFC
0x04	2G Mobile	ThisRFC
0x05	3G Mobile	ThisRFC
0x06	4G Mobile	ThisRFC
0x07	5G Mobile	ThisRFC

Table 2

6.3. New QUIC Network Statistics Registry

A new "QUIC Network Statistics" registry is created. It follows the registration policies defined in [RFC9000], Section 22.1. In addition to the fields described in that section, permanent registrations MUST include the following fields:

Type: The type of statistic, as described in ThisRFC, Section Section 3.

Name: A short name for the field.

The initial value of the table is:

Type	Name
0xc8	Timestamp
0xca	Path Tuple
0xcb	Slow Start Status
0xcc	Network Type
0xcd	Maximum Congestion Window
0xce	Maximum In-Flight Data
0xcf	Smoothed RTT
0xd0	Minimum RTT
0xd1	RTT Variance
0xd2	Latest Bandwidth
0xd3	Maximum Bandwidth
0xd4	Throughput
0xd5	Send Rate
0xd6	Receive Rate
0xd7	Input Rate
0xd8	Loss Rate
0xd9	Buffer Length

Table 3

These fields are permanent, and therefore all have the following values for the common fields:

Field Name	Value
Status	permanent
Specification	ThisRFC
Date	TBD
Change Controller	IETF
Contact	quic@ietf.org
Notes	None

Table 4

7. References

7.1. Normative 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>.

[QUIC-TLS] 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/rfc/rfc9001>.

[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>.

[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/rfc/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[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/rfc/rfc9000>.

[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/rfc/rfc9002>.

7.2. Informative References

[CAREFUL-RESUME] Kuhn, N., Stephan, E., Fairhurst, G., Secchi, R., and C. Huitema, "Convergence of Congestion Control from Retained State", Work in Progress, Internet-Draft, draft-ietf-tsvwg-careful-resume-24, 1 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-careful-resume-24>.

[COOKIES] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/rfc/rfc6265>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/rfc/rfc2104>.

[STUN] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/rfc/rfc5389>.

Appendix A. Sample Integrity Tag implementation

This section is not normative.

We define an Integrity Tag format that consists of a one-byte algorithm identifier, two bytes of private nonce, and a digest. Based on the choice of algorithm, this is a 32-byte SHA256 digest. The entire tag is therefore 35 bytes long.

NOTE Do we need ASCII art for that layout?

The digest will be computed over the nonce, five bytes of 0x01, and the wire-format of the protected fields.

In this example, we will use our third nonce, which is the ASCII values of "quic-new-frame":

```
algid = 1
keyid = 3
nonces[] = [
    [ None ], [ None ],
    [113, 117, 105, 99, 45, 110, 101, 119, 45, 102, 114, 97, 109, 101]
    ]
padding = [ 0x01, 0x01, 0x01, 0x01, 0x01 ]
```

The Network Statistics values are:

NOTE TBD; need a sample network statistics

Which have the following wire representation:

NOTE Calculate them

The value for the Integrity tag is represented by the following psuedo-code:

```
digest = sha56.new()
digest.add(14, nonce[2])
digest.add(5, padding)
digest.add(??, network_statistics)
value = digest.finish()
```

Acknowledgments

TODO acknowledge.

Authors' Addresses

Junghao Yuan
Bytedance
Email: yuanjinghao@bytedance.com

Additional contact information:

袁靖昊
Bytedance

Lei Xiao
Bytedance
Email: xiaolei.shawn@bytedance.com

Additional contact information:

肖磊
Bytedance

Mike Bishop
Akamai Technologies
Email: mbishop@evequefou.be