Stream: Internet Engineering Task Force (IETF)
RFC: 9506
Category: Informational
Published: October 2023
ISSN: 2070-1721
Authors: M. Cociglio, *Telecom Italia - TIM*; A. Ferrieux, *Orange Labs*; G. Fioccola, *Huawei Technologies*; I. Lubashev, *Akamai Technologies*; F. Bulgarella, *Telecom Italia - TIM*; M. Nilo, *Telecom Italia - TIM*; I. Hamchaoui, *Orange Labs*; R. Sisto, *Politecnico di Torino*

RFC 9506
Explicit Host-to-Network Flow Measurements Techniques

Abstract

This document describes protocol-independent methods called Explicit Host-to-Network Flow Measurement Techniques that can be applicable to transport-layer protocols between the client and server. These methods employ just a few marking bits inside the header of each packet for performance measurements and require the client and server to collaborate. Both endpoints cooperate by marking packets and, possibly, mirroring the markings on the round-trip connection. The techniques are especially valuable when applied to protocols that encrypt transport headers since they enable loss and delay measurements by passive, on-path network devices. This document describes several methods that can be used separately or jointly depending of the availability of marking bits, desired measurements, and properties of the protocol to which the methods are applied.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9506.

Copyright Notice

Table of Contents

1. Introduction

Packet loss and delay are hard and pervasive problems of day-to-day network operation. Proactively detecting, measuring, and locating them is crucial to maintaining high QoS and timely resolution of end-to-end throughput issues.

Detecting and measuring packet loss and delay allows network operators to independently confirm trouble reports and, ideally, be proactively notified of developing problems on the network. Locating the cause of packet loss or excessive delay is the first step to resolving problems and restoring QoS.

Network operators wishing to perform quantitative measurement of packet loss and delay have been heavily relying on information present in the clear in transport-layer headers (e.g., TCP sequence and acknowledgment numbers). By passively observing a network path at multiple points within one's network, operators have been able to either quickly locate the source the problem within their network or reliably attribute it to an upstream or downstream network.

With encrypted protocols, the transport-layer headers are encrypted and passive packet loss and delay observations are not possible, as also noted in [TRANSPORT-ENCRYPT]. Nevertheless, accurate measurement of packet loss and delay experienced by encrypted transport-layer protocols is highly desired, especially by network operators who own or control the infrastructure between the client and server.

The measurement of loss and delay experienced by connections using an encrypted protocol cannot be based on a measurement of loss and delay experienced by connections between the same or similar endpoints that use an unencrypted protocol because different protocols may

utilize the network differently and be routed differently by the network. Therefore, it is necessary to directly measure the packet loss and delay experienced by users of encrypted protocols.

The Alternate-Marking method [AltMark] defines a consolidated method to perform packet loss, delay, and jitter measurements on live traffic. However, as mentioned in [IPv6AltMark], [AltMark] mainly applies to a network-layer-controlled domain managed with a Network Management System (NMS), where the Customer Premises Equipment (CPE) or the Provider Edge (PE) routers are the starting or the ending nodes. [AltMark] provides measurement within a controlled domain in which the packets are marked. Therefore, applying [AltMark] to end-to-end transport-layer connections is not easy because packet identification and marking by network nodes is prevented when encrypted transport-layer headers (e.g., QUIC, TCP with TLS) are being used.

This document defines Explicit Host-to-Network Flow Measurement Techniques that are specifically designed for encrypted transport protocols. According to the definitions of [IPPM-METHODS], these measurement methods can be classified as Hybrid. They are to be embedded into a transport-layer protocol and are explicitly intended for exposing delay and loss rate information to on-path measurement devices. Unlike [AltMark], most of these methods require collaborative endpoint nodes. Since these measurement techniques make performance information directly visible to the path, they do not rely on an external NMS.

The Explicit Host-to-Network Flow Measurement Techniques described in this document are applicable to any transport-layer protocol connecting a client and a server. In this document, the client and the server are also referred to as the endpoints of the transport-layer protocol.

The different methods described in this document can be used alone or in combination. Each technique uses few bits and exposes a specific measurement. It is assumed that the endpoints are collaborative in the sense of the measurements, indeed both the client and server need to cooperate.

Following the recommendation in [RFC8558] of making path signals explicit, this document proposes adding some dedicated measurement bits to the clear portion of the transport protocol headers. These bits can be added to an unencrypted portion of a transport-layer header, e.g., UDP surplus space (see [UDP-OPTIONS] and [UDP-SURPLUS]) or reserved bits in a QUIC v1 header, as already done with the latency Spin bit (see Section 17.4 of [QUIC-TRANSPORT]). Note that this document does not recommend the use of any specific bits, as these would need to be chosen by the specific protocol implementations (see Section 5).

The Spin bit, Delay bit, and loss bits explained in this document are inspired by [AltMark], [QUIC-MANAGEABILITY], [QUIC-SPIN], [TSVWG-SPIN], and [IPPM-SPIN].

Additional details about the performance measurements for QUIC are described in the paper [ANRW19-PM-QUIC].

2. Latency Bits

This section introduces bits that can be used for round-trip latency measurements. Whenever this section of the specification refers to packets, it is referring only to packets with protocol headers that include the latency bits.

In [QUIC-TRANSPORT], Section 17.4 introduces an explicit, per-flow transport-layer signal for hybrid measurement of RTT. This signal consists of a Spin bit that toggles once per RTT. Section 4 of [QUIC-SPIN] discusses an additional two-bit Valid Edge Counter (VEC) to compensate for loss and reordering of the Spin bit and to increase fidelity of the signal in less than ideal network conditions.

This document introduces a standalone single-bit delay signal that can be used by passive observers to measure the RTT of a network flow, avoiding the Spin bit ambiguities that arise as soon as network conditions deteriorate.

2.1. Spin Bit

This section is a small recap of the Spin bit working mechanism. For a comprehensive explanation of the algorithm, see Section 3.8.2 of [QUIC-MANAGEABILITY].

The Spin bit is a signal generated by Alternate-Marking [AltMark], where the size of the alternation changes with the flight size each RTT.

The latency Spin bit is a single-bit signal that toggles once per RTT, enabling latency monitoring of a connection-oriented communication from intermediate observation points.

A "Spin bit period" is a set of packets with the same Spin bit value sent during one RTT time interval. A "Spin bit period value" is the value of the Spin bit shared by all packets in a Spin bit period.

The client and server maintain an internal per-connection spin value (i.e., 0 or 1) used to set the Spin bit on outgoing packets. Both endpoints initialize the spin value to 0 when a new connection starts. Then:

- when the client receives a packet with the packet number larger than any number seen so far, it sets the connection spin value to the opposite value contained in the received packet; and
- when the server receives a packet with the packet number larger than any number seen so far, it sets the connection spin value to the same value contained in the received packet.

The computed spin value is used by the endpoints for setting the Spin bit on outgoing packets. This mechanism allows the endpoints to generate a square wave such that, by measuring the distance in time between pairs of consecutive edges observed in the same direction, a passive on-path observer can compute the round-trip network delay of that network flow.

Spin bit enables round-trip latency measurement by observing a single direction of the traffic flow.

Note that packet reordering can cause spurious edges that require heuristics to correct. The Spin bit performance deteriorates as soon as network impairments arise as explained in Section 2.2.

2.2. Delay Bit

The Delay bit has been designed to overcome accuracy limitations experienced by the Spin bit under difficult network conditions:

- packet reordering leads to generation of spurious edges and errors in delay estimation;
- loss of edges causes wrong estimation of Spin bit periods and therefore wrong RTT measurements; and
- application-limited senders cause the Spin bit to measure the application delays instead of network delays.

Unlike the Spin bit, which is set in every packet transmitted on the network, the Delay bit is set only once per round trip.

When the Delay bit is used, a single packet with a marked bit (the Delay bit) bounces between a client and a server during the entire connection lifetime. This single packet is called the "delay sample".

An observer placed at an intermediate point, observing a single direction of traffic and tracking the delay sample and the relative timestamp, can measure the round-trip delay of the connection.

The delay sample lifetime comprises two phases: initialization and reflection. The initialization is the generation of the delay sample, while the reflection realizes the bounce behavior of this single packet between the two endpoints.

The next figure describes the elementary Delay bit mechanism.

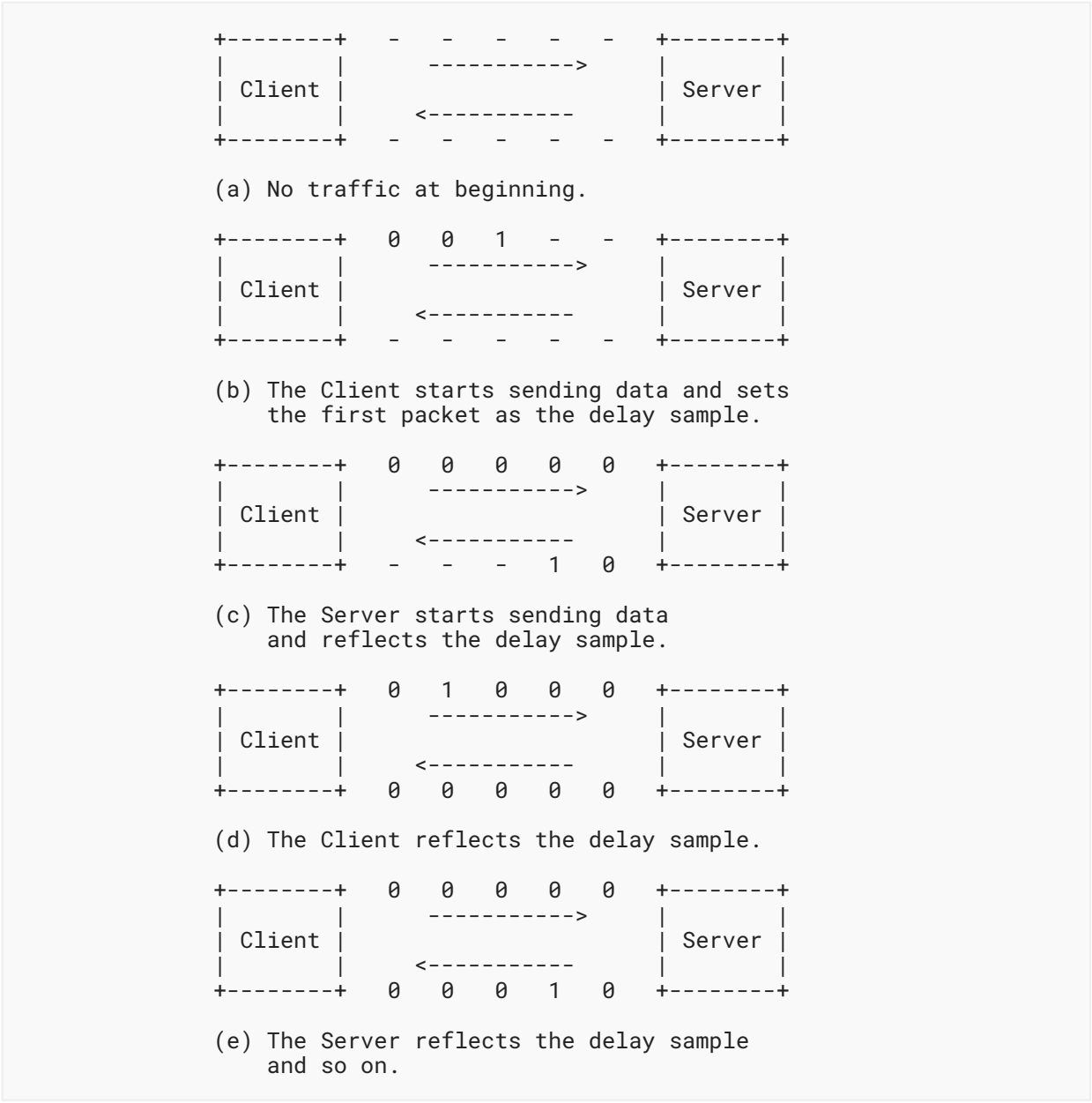

Figure 1: Delay Bit Mechanism

2.2.1. Generation Phase

Only the client is actively involved in the Generation Phase. It maintains an internal per-flow timestamp variable (`ds_time`) updated every time a delay sample is transmitted.

When connection starts, the client generates a new delay sample initializing the Delay bit of the first outgoing packet to 1. Then it updates the `ds_time` variable with the timestamp of its transmission.

The server initializes the Delay bit to 0 at the beginning of the connection, and its only task during the connection is described in Section 2.2.2.

In absence of network impairments, the delay sample should bounce between the client and server continuously for the entire duration of the connection. However, that is highly unlikely for two reasons:

1. The packet carrying the Delay bit might be lost.
2. An endpoint could stop or delay sending packets because the application is limiting the amount of traffic transmitted.

To deal with these problems, the client generates a new delay sample if more than a predetermined time (`T_Max`) has elapsed since the last delay sample transmission (including reflections). Note that `T_Max` should be greater than the max measurable RTT on the network. See Section 2.2.3 for details.

2.2.2. Reflection Phase

Reflection is the process that enables the bouncing of the delay sample between a client and a server. The behavior of the two endpoints is almost the same.

- Server-side reflection: When a delay sample arrives, the server marks the first packet in the opposite direction as the delay sample.
- Client-side reflection: When a delay sample arrives, the client marks the first packet in the opposite direction as the delay sample. It also updates the `ds_time` variable when the outgoing delay sample is actually forwarded.

In both cases, if the outgoing delay sample is being transmitted with a delay greater than a predetermined threshold after the reception of the incoming delay sample (1 ms by default), the delay sample is not reflected, and the outgoing Delay bit is kept at 0.

By doing so, the algorithm can reject measurements that would overestimate the delay due to lack of traffic at the endpoints. Hence, the maximum estimation error would amount to twice the threshold (e.g., 2 ms) per measurement.

2.2.3. `T_Max` Selection

The internal `ds_time` variable allows a client to identify delay sample losses. Considering that a lost delay sample is regenerated at the end of an explicit time (`T_Max`) since the last generation, this same value can be used by an observer to reject a measure and start a new one.

In other words, if the difference in time between two delay samples is greater or equal than `T_Max`, then these cannot be used to produce a delay measure. Therefore, the value of `T_Max` must also be known to the on-path network probes.

There are two alternatives to selecting the T_Max value so that both the client and observers know it. The first one requires that T_Max is known a priori (T_Max_p) and therefore set within the protocol specifications that implements the marking mechanism (e.g., 1 second, which usually is greater than the max expected RTT). The second alternative requires a dynamic mechanism able to adapt the duration of the T_Max to the delay of the connection (T_Max_c).

For instance, the client and observers could use the connection RTT as a basis for calculating an effective T_Max. They should use a predetermined initial value so that T_Max = T_Max_p (e.g., 1 second) and then, when a valid RTT is measured, change T_Max accordingly so that T_Max = T_Max_c. In any case, the selected T_Max should be large enough to absorb any possible variations in the connection delay. This also helps to prevent the mechanism from failing when the observer cannot recognize sudden changes in RTT exceeding T_Max.

T_Max_c could be computed as two times the measured RTT plus a fixed amount of time (100 ms) to prevent low T_Max values in the case of very small RTTs. The resulting formula is: T_Max_c = 2RTT + 100 ms. If T_Max_c is greater than T_Max_p, then T_Max_c is forced to the T_Max_p value. Note that the value of 100 ms is provided as an example, and it may be chosen differently depending on the specific scenarios. For instance, an implementer may consider using existing protocol-specific values if appropriate.

Note that the observer's T_Max should always be less than or equal to the client's T_Max to avoid considering as a valid measurement what is actually the client's T_Max. To obtain this result, the client waits for two consecutive incoming samples and computes the two related RTTs. Then it takes the largest of them as the basis of the T_Max_c formula. At this point, observers have already measured a valid RTT and then computed their T_Max_c.

2.2.4. Delay Measurement Using the Delay Bit

When the Delay bit is used, a passive observer can use delay samples directly and avoid inherent ambiguities in the calculation of the RTT as can be seen in Spin bit analysis.

2.2.4.1. RTT Measurement

The delay sample generation process ensures that only one packet marked with the Delay bit set to 1 runs back and forth between two endpoints per round-trip time. To determine the RTT measurement of a flow, an on-path passive observer computes the time difference between two delay samples observed in a single direction.

To ensure a valid measurement, the observer must verify that the distance in time between the two samples taken into account is less than T_Max.

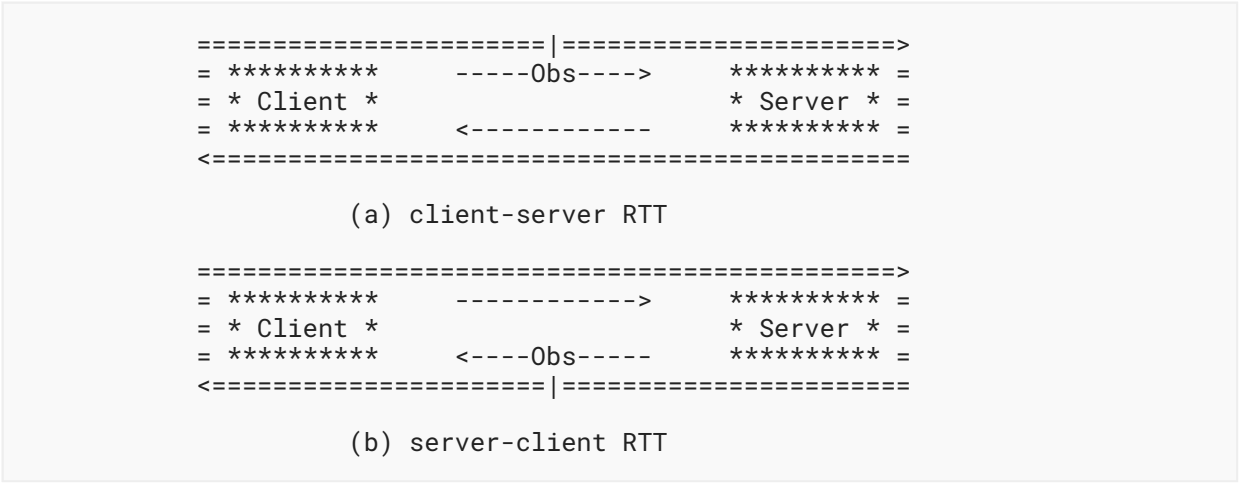

Figure 2: Round-Trip Time (Both Directions)

2.2.4.2. Half-RTT Measurement

An observer that is able to observe both forward and return traffic directions can use the delay samples to measure "upstream" and "downstream" RTT components, also known as the half-RTT measurements. It does this by measuring the time between a delay sample observed in one direction and the delay sample previously observed in the opposite direction.

As with RTT measurement, the observer must verify that the distance in time between the two samples taken into account is less than `T_Max`.

Note that upstream and downstream sections of paths between the endpoints and the observer (i.e., observer-to-client vs. client-to-observer and observer-to-server vs. server-to-observer) may have different delay characteristics due to the difference in network congestion and other factors.

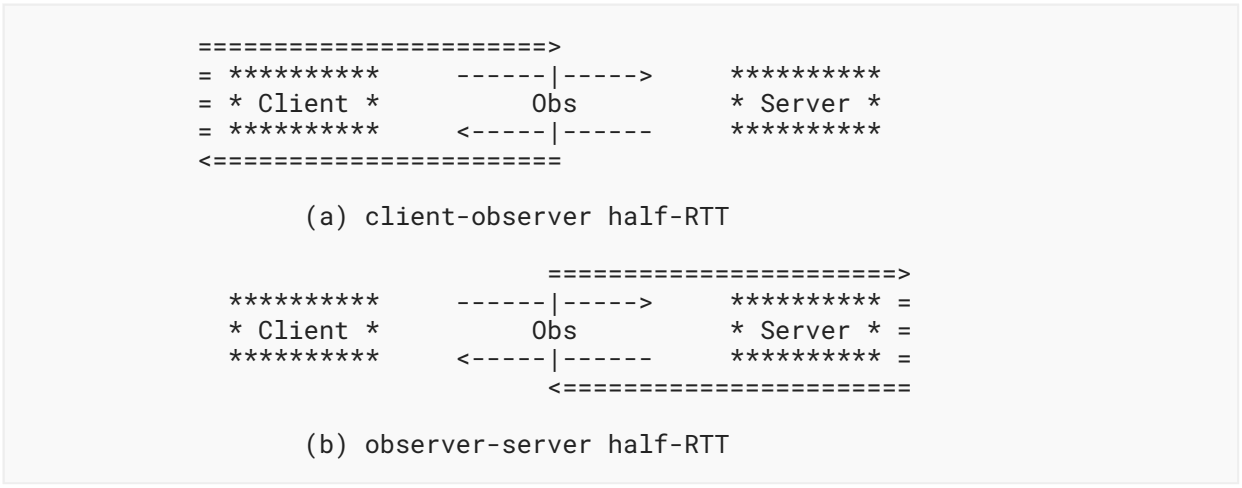

Figure 3: Half Round-Trip Time (Both Directions)

2.2.4.3. Intra-domain RTT Measurement

Intra-domain RTT is the portion of the entire RTT used by a flow to traverse the network of a provider. To measure intra-domain RTT, two observers capable of observing traffic in both directions must be employed simultaneously at the ingress and egress of the network to be measured. Intra-domain RTT is the difference between the two computed upstream (or downstream) RTT components.

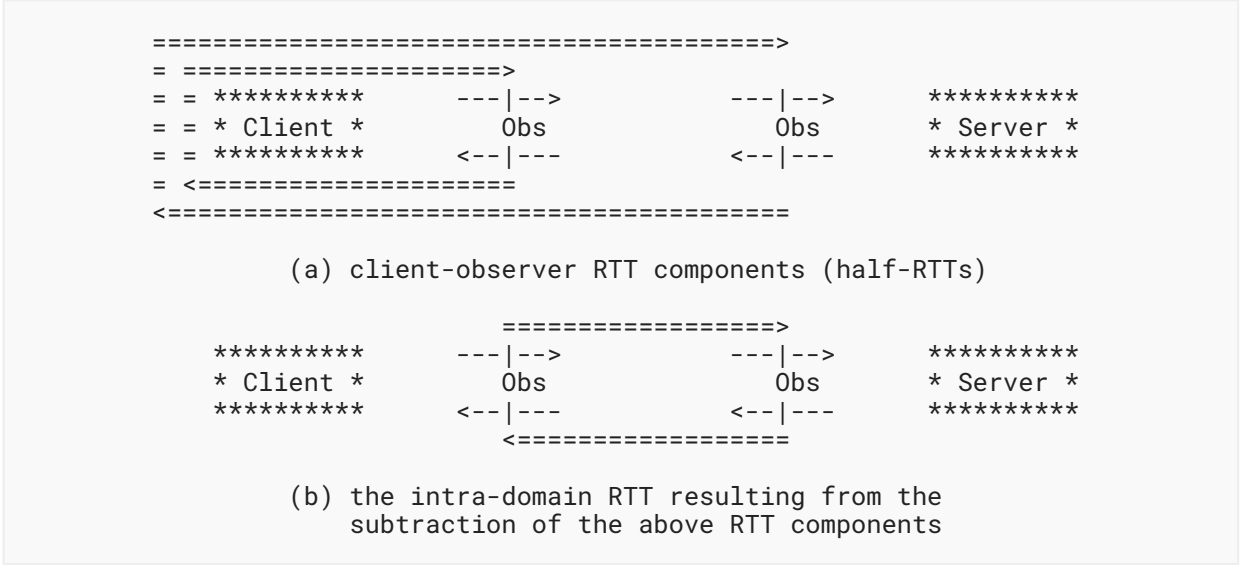

Figure 4: Intra-domain Round-Trip Time (Client-Observer: Upstream)

2.2.5. Observer's Algorithm

An on-path observer maintains an internal per-flow variable to keep track of the time at which the last delay sample has been observed. The flow characterization should be part of the protocol.

If the observer is unidirectional or in case of asymmetric routing, then upon detecting a delay sample:

- if a delay sample was also detected previously in the same direction and the distance in time between them is less than `T_Max - K`, then the two delay samples can be used to calculate RTT measurement. `K` is a protection threshold to absorb differences in `T_Max` computation and delay variations between two consecutive delay samples (e.g., `K = 10% T_Max`).

If the observer can observe both forward and return traffic flows, and it is able to determine which direction contains the client and the server (e.g., by observing the connection handshake), then upon detecting a delay sample:

- if a delay sample was also detected in the opposite direction and the distance in time between them is less than `T_Max - K`, then the two delay samples can be used to measure the observer-client half-RTT or the observer-server half-RTT, according to the direction of the last delay sample observed.

Note that the accuracy can be influenced by what the observer is capable of observing. Additionally, the type of measurement differs, as described in the previous sections.

2.2.6. Two Bits Delay Measurement: Spin Bit + Delay Bit

The Spin and Delay bit algorithms work independently. If both marking methods are used in the same connection, observers can choose the best measurement between the two available:

- when a precise measurement can be produced using the Delay bit, observers choose it; and
- when a Delay bit measurement is not available, observers choose the approximate Spin bit one.

3. Loss Bits

This section introduces bits that can be used for loss measurements. Whenever this section of the specification refers to packets, it is referring only to packets with protocol headers that include the loss bits -- the only packets whose loss can be measured.

T: The "round-Trip loss" bit is used in combination with the Spin bit to measure round-trip loss. See Section 3.1.

Q: The "sQuare" bit is used to measure upstream loss. See Section 3.2.

L: The "Loss Event" bit is used to measure end-to-end loss. See Section 3.3.

R: The "Reflection square" bit is used in combination with the Q bit to measure end-to-end loss. See Section 3.4.

Loss measurements enabled by T, Q, and L bits can be implemented by those loss bits alone (T bit requires a working Spin bit). Two-bit combinations Q+L and Q+R enable additional measurement opportunities discussed below.

Each endpoint maintains appropriate counters independently and separately for each identifiable flow (or each sub-flow for multipath connections).

Since loss is reported independently for each flow, all bits (except for the L bit) require a certain minimum number of packets to be exchanged per flow before any signal can be measured. Therefore, loss measurements work best for flows that transfer more than a minimal amount of data.

3.1. T Bit -- Round-Trip Loss Bit

The round-Trip loss bit is used to mark a variable number of packets exchanged twice between the endpoints realizing a two round-trip reflection. A passive on-path observer, observing either direction, can count and compare the number of marked packets seen during the two reflections, estimating the loss rate experienced by the connection. The overall exchange comprises:

- the client selects and consequently sets the T bit to 1 in order to identify a first train of packets;
- upon receiving each packet included in the first train, the server sets the T bit to 1 and reflects to the client a respective second train of packets of the same size as the first train received;
- upon receiving each packet included in the second train, the client sets the T bit to 1 and reflects to the server a respective third train of packets of the same size as the second train received; and
- upon receiving each packet included in the third train, the server sets the T bit to 1 and finally reflects to the client a respective fourth train of packets of the same size as the third train received.

Packets belonging to the first round trip (first and second train) represent the Generation Phase, while those belonging to the second round trip (third and fourth train) represent the Reflection Phase.

A passive on-path observer can count and compare the number of marked packets seen during the two round trips (i.e., the first and third or the second and the fourth trains of packets, depending on which direction is observed) and estimate the loss rate experienced by the connection. This process is repeated continuously to obtain more measurements as long as the endpoints exchange traffic. These measurements can be called round-trip losses.

Since the packet rates in two directions may be different, the number of marked packets in the train is determined by the direction with the lowest packet rate. See Section 3.1.2 for details on packet generation.

3.1.1. Round-Trip Loss

Since the measurements are performed on a portion of the traffic exchanged between the client and the server, the observer calculates the end-to-end Round-Trip Packet Loss (RTPL) that, statistically, will correspond to the loss rate experienced by the connection along the entire network path.

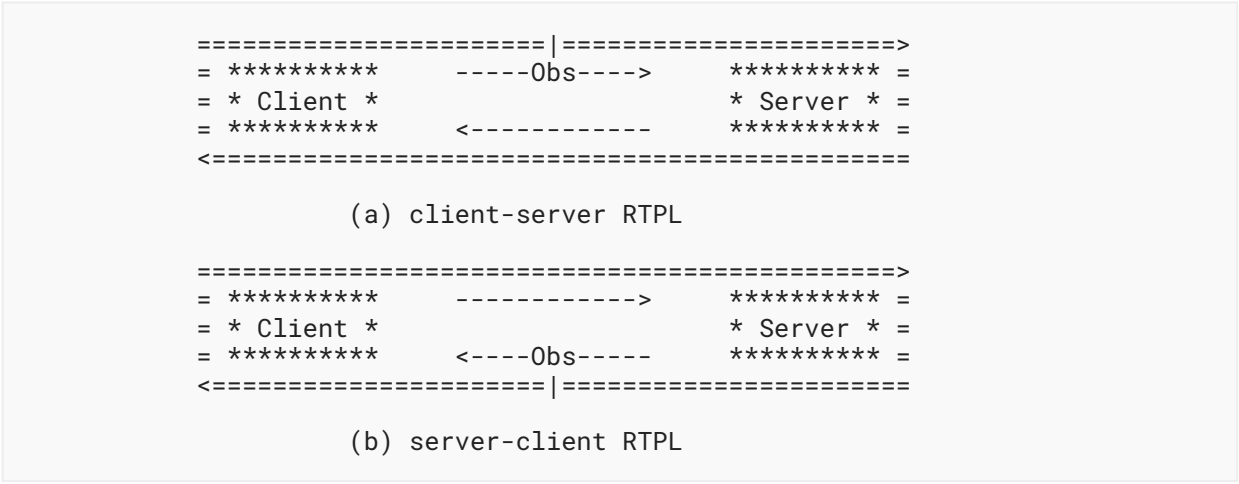

Figure 5: Round-Trip Packet Loss (Both Directions)

This methodology also allows the half-RTPL measurement and the Intra-domain RTPL measurement in a way similar to RTT measurement.

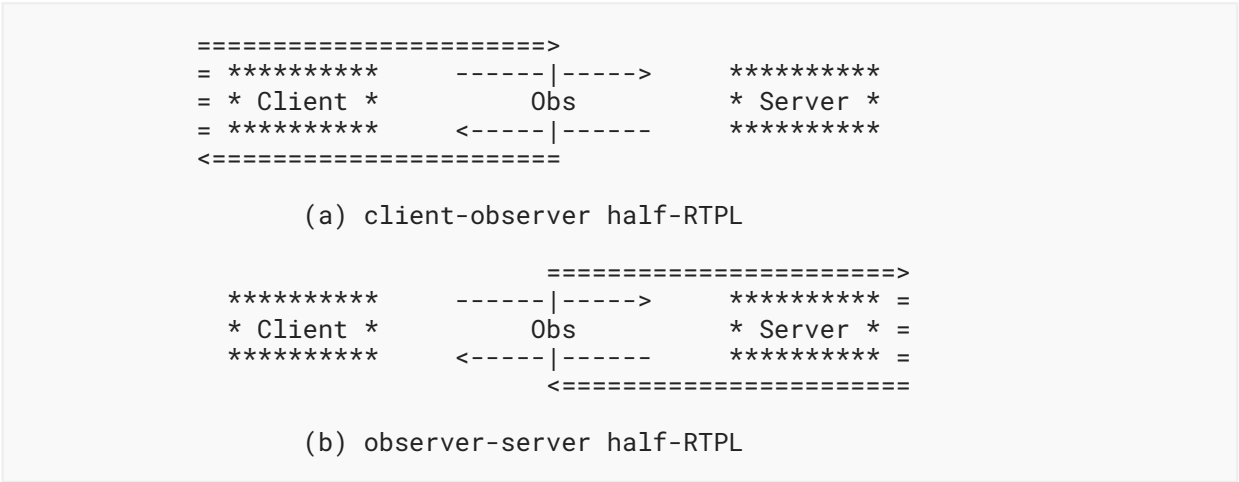

Figure 6: Half Round-Trip Packet Loss (Both Directions)

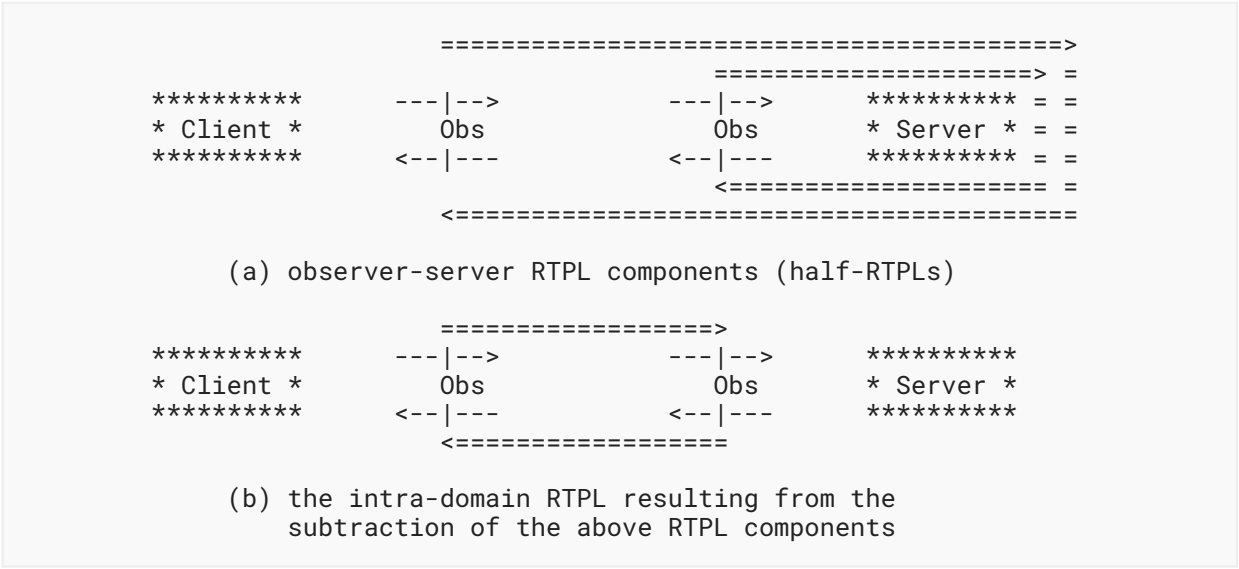

Figure 7: Intra-domain Round-Trip Packet Loss (Observer-Server)

3.1.2. Setting the Round-Trip Loss Bit on Outgoing Packets

The round-Trip loss signal requires a working Spin bit signal to separate trains of marked packets (packets with T bit set to 1). A "pause" of at least one empty Spin bit period between each phase of the algorithm serves as such a separator for the on-path observer. The connection between T bit and Spin bit helps the observer correlate packet trains.

The client maintains a "generation token" count that is set to zero at the beginning of the session and is incremented every time a packet is received (marked or unmarked). The client also maintains a "reflection counter" that starts at zero at the beginning of the session.

The client is in charge of launching trains of marked packets and does so according to the algorithm:

1. Generation Phase. The client starts generating marked packets for two consecutive Spin bit periods. When the client transmits a packet and a "generation token" is available, the client marks the packet and retires a "generation token". If no token is available, the outgoing packet is transmitted unmarked. At the end of the first Spin bit period spent in generation, the reflection counter is unlocked to start counting incoming marked packets that will be reflected later.
2. Pause Phase. When the generation is completed, the client pauses till it has observed one entire Spin bit period with no marked packets. That Spin bit period is used by the observer as a separator between generated and reflected packets. During this marking pause, all the outgoing packets are transmitted with T bit set to 0. The reflection counter is still incremented every time a marked packet arrives.
3. Reflection Phase. The client starts transmitting marked packets, decrementing the reflection counter for each transmitted marked packet until the reflection counter has reached zero. The "generation token" method from the Generation Phase is used during this phase as well.

At the end of the first Spin bit period spent in reflection, the reflection counter is locked to avoid incoming reflected packets incrementing it.

4. Pause Phase 2. The Pause Phase is repeated after the Reflection Phase and serves as a separator between the reflected packet train and a new packet train.

The generation token counter should be capped to limit the effects of a subsequent sudden reduction in the other endpoint's packet rate that could prevent that endpoint from reflecting collected packets. A cap value of 1 is recommended.

A server maintains a "marking counter" that starts at zero and is incremented every time a marked packet arrives. When the server transmits a packet and the "marking counter" is positive, the server marks the packet and decrements the "marking counter". If the "marking counter" is zero, the outgoing packet is transmitted unmarked.

Note that a choice of 2 RTT (two Spin bit periods) for the Generation Phase is a trade-off between the percentage of marked packets (i.e., the percentage of traffic monitored) and the measurement delay. Using this value, the algorithm produces a measurement approximately every 6 RTT (2 generations, ~2 reflections, 2 pauses), marking ~1/3 of packets exchanged in the slower direction (see Section 3.1.4). Choosing a Generation Phase of 1 RTT, we would produce measurements every 4 RTT, monitoring ~1/4 of packets in the slower direction.

It is worth mentioning that problems can happen in some cases, especially if the rate suddenly changes, but the mechanism described here worked well with normal traffic conditions in the implementation.

3.1.3. Observer's Logic for Round-Trip Loss Signal

The on-path observer counts marked packets and separates different trains by detecting Spin bit periods (at least one) with no marked packets. The Round-Trip Packet Loss (RTPL) is the difference between the size of the Generation train and the Reflection train.

In the following example, packets are represented by two bits (first one is the Spin bit, second one is the round-Trip loss bit):

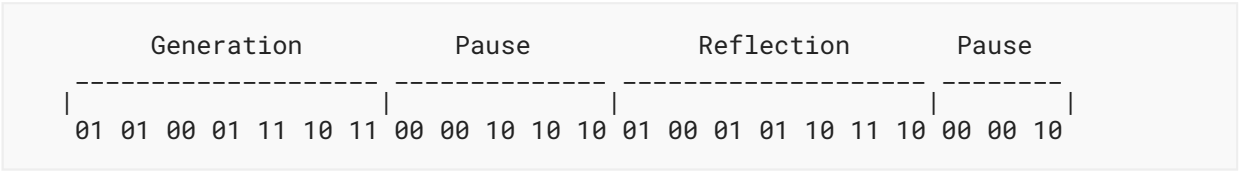

Figure 8: Round-Trip Loss Signal Example

Note that 5 marked packets have been generated, of which 4 have been reflected.

3.1.4. Loss Coverage and Signal Timing

A cycle of the round-Trip loss signaling algorithm contains 2 RTTs of Generation phase, 2 RTTs of Reflection Phase, and 2 Pause Phases at least 1 RTT in duration each. Hence, the loss signal is delayed by about 6 RTTs since the loss events.

The observer can only detect the loss of marked packets that occurs after its initial observation of the Generation Phase and before its subsequent observation of the Reflection Phase. Hence, if the loss occurs on the path that sends packets at a lower rate (typically ACKs in such asymmetric scenarios), 2/6 (1/3) of the packets will be sampled for loss detection.

If the loss occurs on the path that sends packets at a higher rate, `lowPacketRate/(3*highPacketRate)` of the packets will be sampled for loss detection. For protocols that use ACKs, the portion of packets sampled for loss in the higher rate direction during unidirectional data transfer is `1/(3*packetsPerAck)`, where the value of `packetsPerAck` can vary by protocol, by implementation, and by network conditions.

3.2. Q Bit -- sQuare Bit

The sQuare bit (Q bit) takes its name from the square wave generated by its signal. This method is based on the Alternate-Marking method [AltMark], and the Q bit represents the "packet color" that can be switched between 0 and 1 in order to mark consecutive blocks of packets with different colors. This method does not require cooperation from both endpoints.

[AltMark] introduces two variations of the Alternate-Marking method depending on whether the color is switched according to a fixed timer or after a fixed number of packets. Cooperating and synchronized observers on either end of a network segment can use the fixed-timer method to measure packet loss on the segment by comparing packet counters for the same packet blocks. The time length of the blocks can be chosen depending on the desired measurement frequency, but it must be long enough to guarantee the proper operation with respect to clock errors and network delay issues.

The Q bit method described in this document chooses the color-switching method based on a fixed number of packets for each block. This approach has the advantage that it does not require cooperating or synchronized observers or network elements. Each probe can measure packet loss autonomously without relying on an external NMS. For the purpose of the packet loss measurement, all blocks have the same number of packets, and it is necessary to detect only the loss event and not to identify the exact block with losses.

Following the method based on fixed number of packets, the square wave signal is generated by the switching of the Q bit: every outgoing packet contains the Q bit value, which is initialized to 0 and inverted after sending N packets (a sQuare Block or simply Q Block). Hence, Q Period is 2*N.

Observation points can estimate upstream losses by watching a single direction of the traffic flow and counting the number of packets in each observed Q Block, as described in Section 3.2.2.

3.2.1. Q Block Length Selection

The length of the block must be known to the on-path network probes. There are two alternatives to selecting the Q Block length. The first one requires that the length is known a priori and therefore set within the protocol specifications that implement the marking mechanism. The second requires the sender to select it.

In this latter scenario, the sender is expected to choose N (Q Block length) based on the expected amount of loss and reordering on the path. The choice of N strikes a compromise -- the observation could become too unreliable in case of packet reordering and/or severe loss if N is too small, while short flows may not yield a useful upstream loss measurement if N is too large (see Section 3.2.2).

The value of N should be at least 64 and be a power of 2. This requirement allows an observer to infer the Q Block length by observing one period of the square signal. It also allows the observer to identify flows that set the loss bits to arbitrary values (see Section 6).

If the sender does not have sufficient information to make an informed decision about Q Block length, the sender should use N=64, since this value has been extensively tried in large-scale field tests and yielded good results. Alternatively, the sender may also choose a random power-of-2 N for each flow, increasing the chances of using a Q Block length that gives the best signal for some flows.

The sender must keep the value of N constant for a given flow.

3.2.2. Upstream Loss

Blocks of N (Q Block length) consecutive packets are sent with the same value of the Q bit, followed by another block of N packets with an inverted value of the Q bit. Hence, knowing the value of N, an on-path observer can estimate the amount of upstream loss after observing at least N packets. The upstream loss rate (`uloss`) is one minus the average number of packets in a block of packets with the same Q value (`p`) divided by N (`uloss=1-avg(p)/N`).

The observer needs to be able to tolerate packet reordering that can blur the edges of the square signal, as explained in Section 3.2.3.

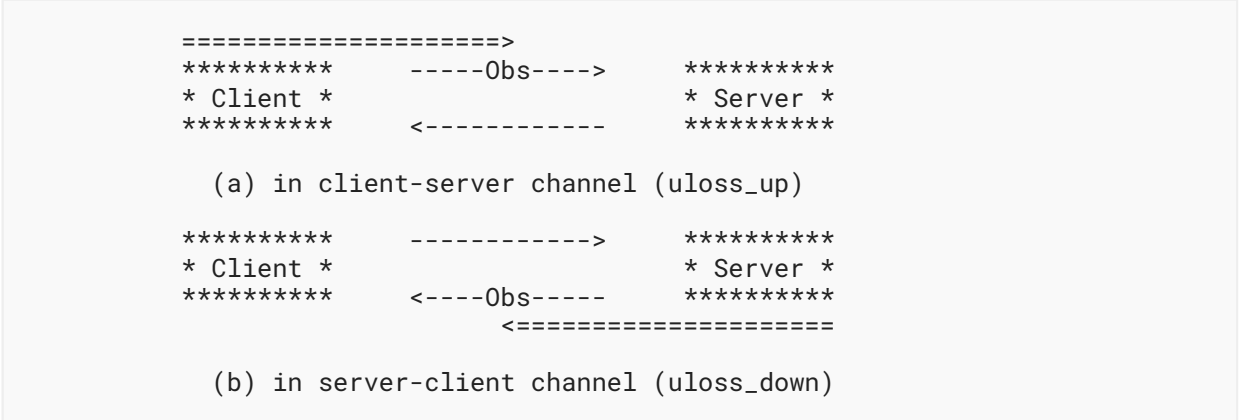

Figure 9: Upstream Loss

3.2.3. Identifying Q Block Boundaries

Packet reordering can produce spurious edges in the square signal. To address this, the observer should look for packets with the current Q bit value up to X packets past the first packet with a reverse Q bit value. The value of X, a "Marking Block Threshold", should be less than `N/2`.

The choice of X represents a trade-off between resiliency to reordering and resiliency to loss. A very large Marking Block Threshold will be able to reconstruct Q Blocks despite a significant amount of reordering, but it may erroneously coalesce packets from multiple Q Blocks into fewer Q Blocks if loss exceeds 50% for some Q Blocks.

3.2.3.1. Improved Resilience to Burst Losses

Burst losses can affect the accuracy of Q measurements. Generally, burst losses can be absorbed and correctly measured if smaller than the established Q Block length. If the entire Q Block length of packets is lost in a burst, however, the observer may be left completely unaware of the loss.

To improve burst loss resilience, an observer may consider a received Q Block larger than the selected Q Block length as an indication of a burst loss event. The observer would then compute the loss as three times the Q Block length minus the measured block length. By doing so, the observer can detect burst losses of less than two blocks (e.g., less than 128 packets for a Q Block length of 64 packets). A burst loss of two or more consecutive periods would still remain unnoticed by the observer (or underestimated if a period longer than Q Block length were formed).

3.3. L Bit -- Loss Event Bit

The Loss Event bit uses an Unreported Loss counter maintained by the protocol that implements the marking mechanism. To use the Loss Event bit, the protocol must allow the sender to identify lost packets. This is true of protocols such as QUIC, partially true for TCP and Stream Control Transmission Protocol (SCTP) (losses of pure ACKs are not detected), and is not true of protocols such as UDP and IPv4/IPv6.

The Unreported Loss counter is initialized to 0, and the L bit of every outgoing packet indicates whether the Unreported Loss counter is positive (L=1 if the counter is positive, and L=0 otherwise).

The value of the Unreported Loss counter is decremented every time a packet with L=1 is sent.

The value of the Unreported Loss counter is incremented for every packet that the protocol declares lost, using whatever loss detection machinery the protocol employs. If the protocol is able to rescind the loss determination later, a positive Unreported Loss counter may be decremented due to the rescission. In general, it should not become negative due to the rescission, but it can happen in few cases.

This loss signaling is similar to loss signaling in [ConEx], except that the Loss Event bit is reporting the exact number of lost packets, whereas the signal mechanism in [ConEx] is reporting an approximate number of lost bytes.

For protocols, such as TCP [TCP], that allow network devices to change data segmentation, it is possible that only a part of the packet is lost. In these cases, the sender must increment the Unreported Loss counter by the fraction of the packet data lost (so the Unreported Loss counter may become negative when a packet with L=1 is sent after a partial packet has been lost).

Observation points can estimate the end-to-end loss, as determined by the upstream endpoint, by counting packets in this direction with the L bit equal to 1, as described in Section 3.3.1.

3.3.1. End-To-End Loss

The Loss Event bit allows an observer to estimate the end-to-end loss rate by counting packets with L bit values of 0 and 1 for a given flow. The end-to-end loss ratio is the fraction of packets with L=1.

The assumption here is that upstream loss affects packets with L=0 and L=1 equally. If some loss is caused by tail-drop in a network device, this may be a simplification. If the sender's congestion controller reduces the packet send rate after loss, there may be a sufficient delay before sending packets with L=1 that they have a greater chance of arriving at the observer.

3.3.1.1. Loss Profile Characterization

The Loss Event bit allows an observer to characterize the loss profile, since the distribution of observed packets with the L bit set to 1 roughly corresponds to the distribution of packets lost between 1 RTT and 1 retransmission timeout (RTO) before (see Section 3.3.2.1). Hence, observing random single instances of the L bit set to 1 indicates random single packet loss, while observing blocks of packets with the L bit set to 1 indicates loss affecting entire blocks of packets.

3.3.2. L+Q Bits -- Loss Measurement Using L and Q Bits

Combining L and Q bits allows a passive observer watching a single direction of traffic to accurately measure:

upstream loss: sender-to-observer loss (see Section 3.2.2)

downstream loss: observer-to-receiver loss (see Section 3.3.2.2)

end-to-end loss: sender-to-receiver loss on the observed path (see Section 3.3.1) with loss profile characterization (see Section 3.3.1.1)

3.3.2.1. Correlating End-to-End and Upstream Loss

Upstream loss is calculated by observing packets that did not suffer the upstream loss (Section 3.2.2). End-to-end loss, however, is calculated by observing subsequent packets after the sender's protocol detected the loss. Hence, end-to-end loss is generally observed with a delay of between 1 RTT (loss declared due to multiple duplicate acknowledgments) and 1 RTO (loss declared due to a timeout) relative to the upstream loss.

The flow RTT can sometimes be estimated by timing the protocol handshake messages. This RTT estimate can be greatly improved by observing a dedicated protocol mechanism for conveying RTT information, such as the Spin bit (see Section 2.1) or Delay bit (see Section 2.2).

Whenever the observer needs to perform a computation that uses both upstream and end-to-end loss rate measurements, it should consider the upstream loss rate leading the end-to-end loss rate by approximately 1 RTT. If the observer is unable to estimate RTT of the flow, it should accumulate loss measurements over time periods of at least 4 times the typical RTT for the observed flows.

If the calculated upstream loss rate exceeds the end-to-end loss rate calculated in Section 3.3.1, then either the Q Period is too short for the amount of packet reordering or there is observer loss, described in Section 3.3.2.3. If this happens, the observer should adjust the calculated upstream loss rate to match end-to-end loss rate, unless the following applies.

In case of a protocol, such as TCP or SCTP, that does not track losses of pure ACK packets, observing a direction of traffic dominated by pure ACK packets could result in measured upstream loss that is higher than measured end-to-end loss if said pure ACK packets are lost upstream. Hence, if the measurement is applied to such protocols, and the observer can confirm that pure ACK packets dominate the observed traffic direction, the observer should adjust the calculated end-to-end loss rate to match upstream loss rate.

3.3.2.2. Downstream Loss

Because downstream loss affects only those packets that did not suffer upstream loss, the end-to-end loss rate (`eloss`) relates to the upstream loss rate (`uloss`) and downstream loss rate (`dloss`) as `(1-uloss)(1-dloss)=1-eloss`. Hence, `dloss=(eloss-uloss)/(1-uloss)`.

3.3.2.3. Observer Loss

A typical deployment of a passive observation system includes a network tap device that mirrors network packets of interest to a device that performs analysis and measurement on the mirrored packets. The observer loss is the loss that occurs on the mirror path.

Observer loss affects the upstream loss rate measurement since it causes the observer to account for fewer packets in a block of identical Q bit values (see Section 3.2.2). The end-to-end loss rate measurement, however, is unaffected by the observer loss since it is a measurement of the fraction of packets with the L bit value of 1, and the observer loss would affect all packets equally (see Section 3.3.1).

The need to adjust the upstream loss rate down to match the end-to-end loss rate as described in Section 3.3.2.1 is an indication of the observer loss, whose magnitude is between the amount of such adjustment and the entirety of the upstream loss measured in Section 3.2.2. Alternatively, a high apparent upstream loss rate could be an indication of significant packet reordering, possibly due to packets belonging to a single flow being multiplexed over several upstream paths with different latency characteristics.

3.4. R Bit -- Reflection Square Bit

R bit requires a deployment alongside Q bit. Unlike the square signal for which packets are transmitted in blocks of fixed size, the number of packets in Reflection square blocks (also an Alternate-Marking signal) varies according to these rules:

- when the transmission of a new block starts, its size is set equal to the size of the last Q Block whose reception has been completed; and
- if the reception of at least one further Q Block is completed before transmission of the block is terminated, the size of the block is updated to be the average size of the further received Q Blocks.

The Reflection square value is initialized to 0 and is applied to the R bit of every outgoing packet. The Reflection square value is toggled for the first time when the completion of a Q Block is detected in the incoming square signal (produced by the other endpoint using the Q bit). The number of packets detected within this first Q Block (`p`), is used to generate a reflection square signal that toggles every `M=p` packets (at first). This new signal produces blocks of M packets (marked using the R bit) and each of them is called "Reflection Block" (Reflection Block).

The M value is then updated every time a completed Q Block in the incoming square signal is received, following this formula: `M=round(avg(p))`.

The parameter `avg(p)`, the average number of packets in a marking period, is computed based on all the Q Blocks received since the beginning of the current Reflection Block.

The transmission of a Reflection Block is considered complete (and the signal toggled) when the number of packets transmitted in that block is at least the latest computed M value.

To ensure a proper computation of the M value, endpoints implementing the R bit must identify the boundaries of incoming Q Blocks. The same approach described in Section 3.2.3 should be used.

By looking at the R bit, unidirectional observation points have an indication of loss experienced by the entire unobserved channel plus the loss on the path from the sender to them.

Since the Q Block is sent in one direction, and the corresponding reflected R Block is sent in the opposite direction, the reflected R signal is transmitted with the packet rate of the slowest direction. Namely, if the observed direction is the slowest, there can be multiple Q Blocks transmitted in the unobserved direction before a complete Reflection Block is transmitted in the observed direction. If the unobserved direction is the slowest, the observed direction can be sending R Blocks of the same size repeatedly before it can update the signal to account for a newly completed Q Block.

3.4.1. Enhancement of Reflection Block Length Computation

The use of the rounding function used in the M computation introduces errors that can be minimized by storing the rounding applied each time M is computed and using it during the computation of the M value in the following Reflection Block.

This can be achieved by introducing the new `r_avg` parameter in the computation of M. The new formula is `Mr=avg(p)+r_avg; M=round(Mr); r_avg=Mr-M` where the initial value of `r_avg` is equal to 0.

3.4.2. Improved Resilience to Packet Reordering

When a protocol implementing the marking mechanism is able to detect when packets are received out of order, it can improve resilience to packet reordering beyond what is possible by using methods described in Section 3.2.3.

This can be achieved by updating the size of the current Reflection Block while it is being transmitted. The Reflection Block size is then updated every time an incoming reordered packet of the previous Q Block is detected. This can be done if and only if the transmission of the current Reflection Block is in progress and no packets of the following Q Block have been received.

3.4.2.1. Improved Resilience to Burst Losses

Burst losses can affect the accuracy of R measurements similar to how they affect accuracy of Q measurements. Therefore, recommendations in Section 3.2.3.1 apply equally to improving burst loss resilience for R measurements.

3.4.3. R+Q Bits -- Loss Measurement Using R and Q Bits

Since both sQuare and Reflection square bits are toggled at most every N packets (except for the first transition of the R bit as explained before), an on-path observer can count the number of packets of each marking block and, knowing the value of N, can estimate the amount of loss experienced by the connection. An observer can calculate different measurements depending on whether it is able to observe a single direction of the traffic or both directions.

Single directional observer:

upstream loss in the observed direction: the loss between the sender and the observation point (see Section 3.2.2)

"three-quarters" connection loss: the loss between the receiver and the sender in the unobserved direction plus the loss between the sender and the observation point in the observed direction

end-to-end loss in the unobserved direction: the loss between the receiver and the sender in the opposite direction

Two directions observer (same metrics seen previously applied to both direction, plus):

client-observer half round-trip loss: the loss between the client and the observation point in both directions

observer-server half round-trip loss: the loss between the observation point and the server in both directions

downstream loss: the loss between the observation point and the receiver (applicable to both directions)

3.4.3.1. Three-Quarters Connection Loss

Except for the very first block in which there is nothing to reflect (a complete Q Block has not been yet received), packets are continuously R-bit marked into alternate blocks of size lower or equal than N. By knowing the value of N, an on-path observer can estimate the amount of loss that has occurred in the whole opposite channel plus the loss from the sender up to it in the observation channel. As for the previous metric, the `three-quarters` connection loss rate (`tqloss`) is one minus the average number of packets in a block of packets with the same R value (`t`) divided by `N` (`tqloss=1-avg(t)/N`).

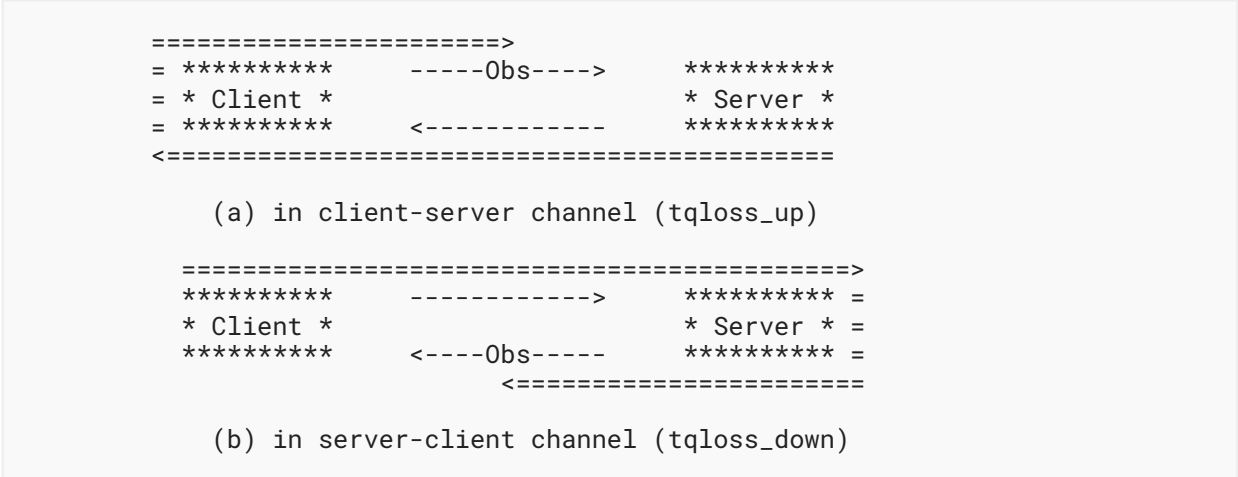

Figure 10: Three-Quarters Connection Loss

The following metrics derive from this last metric and the upstream loss produced by the Q bit.

3.4.3.2. End-To-End Loss in the Opposite Direction

End-to-end loss in the unobserved direction (`eloss_unobserved`) relates to the "three-quarters" connection loss (`tqloss`) and upstream loss in the observed direction (`uloss`) as `(1-eloss_unobserved)(1-uloss)=1-tqloss`. Hence, `eloss_unobserved=(tqloss-uloss)/(1-uloss)`.

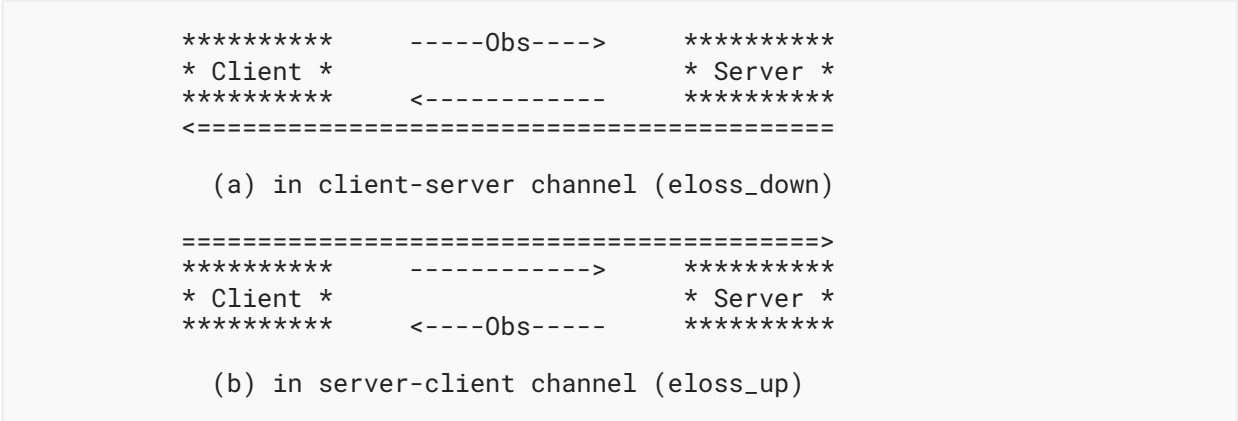

Figure 11: End-To-End Loss in the Opposite Direction

3.4.3.3. Half Round-Trip Loss

If the observer is able to observe both directions of traffic, it is able to calculate two "half round-trip" loss measurements -- loss from the observer to the receiver (in a given direction) and then back to the observer in the opposite direction. For both directions, "half round-trip" loss (`hrtloss`) relates to "three-quarters" connection loss (`tqloss_opposite`) measured in the opposite direction and the upstream loss (`uloss`) measured in the given direction as `(1-uloss)(1-hrtloss)=1-tqloss_opposite`. Hence, `hrtloss=(tqloss_opposite-uloss)/(1-uloss)`.

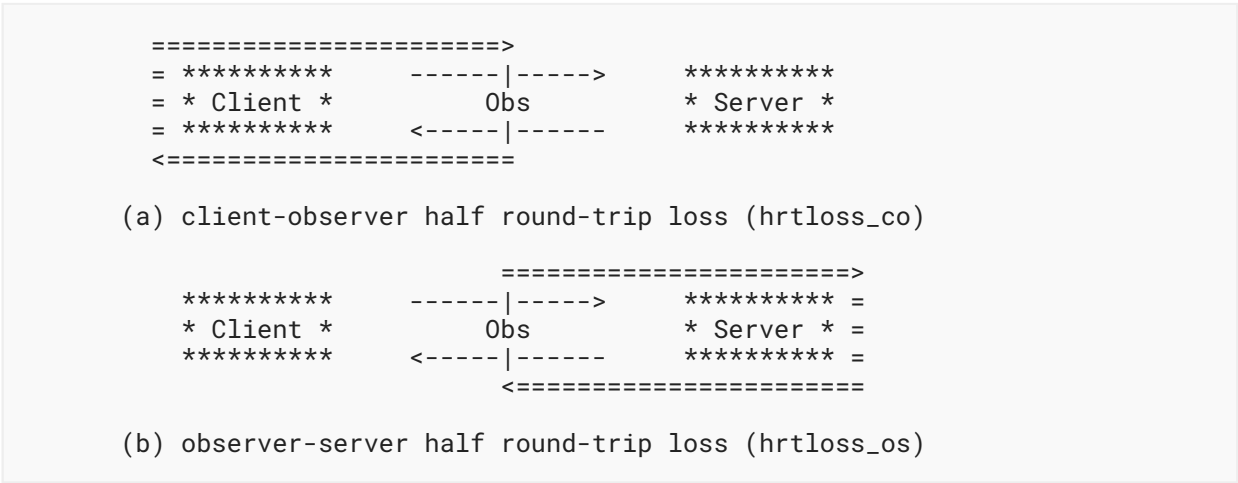

Figure 12: Half Round-Trip Loss (Both Directions)

3.4.3.4. Downstream Loss

If the observer is able to observe both directions of traffic, it is able to calculate two downstream loss measurements using either end-to-end loss and upstream loss, similar to the calculation in Section 3.3.2.2, or "half round-trip" loss and upstream loss in the opposite direction.

For the latter, `dloss=(hrtloss-uloss_opposite)/(1-uloss_opposite)`.

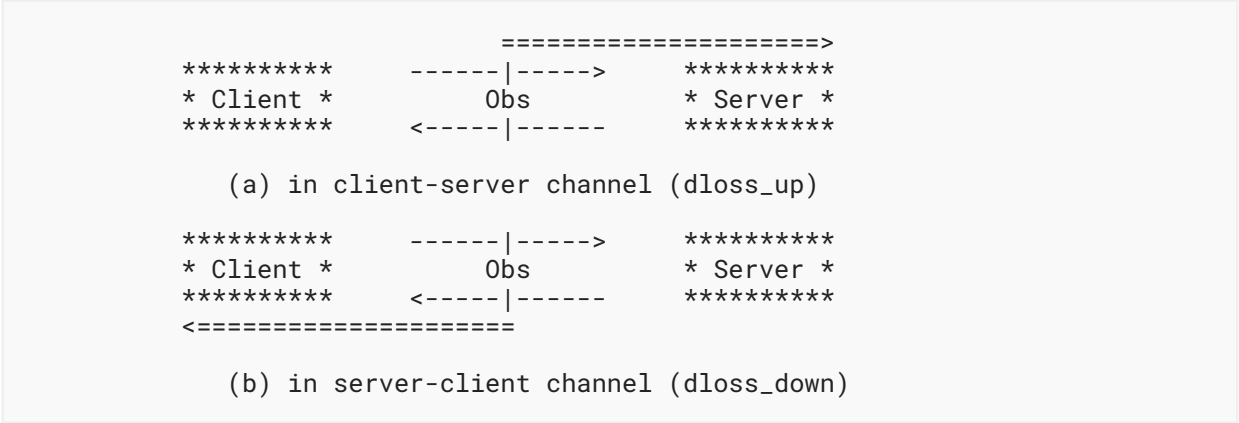

Figure 13: Downstream Loss

3.5. E Bit -- ECN-Echo Event Bit

While the primary focus of this document is on exposing packet loss and delay, modern networks can report congestion before they are forced to drop packets, as described in [ECN]. When transport protocols keep ECN-Echo feedback under encryption, this signal cannot be observed by the network operators. When tasked with diagnosing network performance problems, knowledge of a congestion downstream of an observation point can be instrumental.

If downstream congestion information is desired, this information can be signaled with an additional bit.

E: The "ECN-Echo Event" bit is set to 0 or 1 according to the Unreported ECN-Echo counter, as explained below in Section 3.5.1.

3.5.1. Setting the ECN-Echo Event Bit on Outgoing Packets

The Unreported ECN-Echo counter operates identically to Unreported Loss counter (Section 3.3), except it counts packets delivered by the network with Congestion Experienced (CE) markings, according to the ECN-Echo feedback from the receiver.

This ECN-Echo signaling is similar to ECN signaling in [ConEx]. The ECN-Echo mechanism in QUIC provides the number of packets received with CE marks. For protocols like TCP, the method described in [ConEx-TCP] can be employed. As stated in [ConEx-TCP], such feedback can be further improved using a method described in [ACCURATE-ECN].

3.5.2. Using E Bit for Passive ECN-Reported Congestion Measurement

A network observer can count packets with the CE codepoint and determine the upstream CE-marking rate directly.

Observation points can also estimate ECN-reported end-to-end congestion by counting packets in this direction with an E bit equal to 1.

The upstream CE-marking rate and end-to-end ECN-reported congestion can provide information about the downstream CE-marking rate. The presence of E bits along with L bits, however, can somewhat confound precise estimates of upstream and downstream CE markings if the flow contains packets that are not ECN capable.

3.5.3. Multiple E Bits

Some protocols, such as QUIC, support separate ECN-Echo counters. For example, Section 13.4.1 of [QUIC-TRANSPORT] describes separate counters for ECT(0), ECT(1), and ECN-CE. To better support such protocols, multiple E bits can be used, one per a corresponding ECN-Echo counter.

4. Summary of Delay and Loss Marking Methods

This section summarizes the marking methods described in this document, which proposes a toolkit of techniques that can be used separately, partly, or all together depending on the need.

For the delay measurement, it is possible to use the Spin bit and/or the Delay bit. A unidirectional or bidirectional observer can be used.

Method	# of bits	Available Delay Metrics		Impairments Resiliency	# of meas.
		UniDir Observer	BiDir Observer		
S: Spin Bit	1	RTT	x2, Half RTT	low	very high
D: Delay Bit	1	RTT	x2, Half RTT	high	medium
SD: Spin Bit & Delay Bit *	2	RTT	x2, Half RTT	high	very high

Table 1: Delay Comparison

x2 Same metric for both directions

* Both bits work independently; an observer could use less accurate Spin bit measurements when Delay bit ones are unavailable.

For the Loss measurement, each row in Table 2 represents a loss-marking method. For each method, the table specifies the number of bits required in the header, the available metrics using a unidirectional or bidirectional observer, applicable protocols, measurement fidelity, and delay.

Method	Bits	Available Loss Metrics		Prto	Measurement Aspects	
		UniDir Observer	BiDir Observer		Fidelity	Delay
T: Round-Trip Loss Bit	$1	RT	x2, Half RT	*	Rate by sampling 1/3 to 1/(3*ppa) of pkts over 2 RTT	~6 RTT
Q: sQuare Bit	1	Upstream	x2	*	Rate over N pkts (e.g., 64)	N pkts (e.g., 64)
L: Loss Event Bit	1	E2E	x2	#	Loss shape (and rate)	Min: RTT, Max: RTO
QL: sQuare + Loss Ev. Bits	2	Upstream	x2	#	see Q	see Q
		Downstream	x2	#	see Q|L	see L
		E2E	x2	#	see L	see L
QR: sQuare + Ref. Sq. Bits	2	Upstream	x2	*	Rate over N*ppa pkts (see Q bit for N)	see Q
		3/4 RT	x2	*		N*ppa pkts (see Q bit for N)
		!E2E	E2E, Downstream, Half RT	*		

Table 2: Loss Comparison

*	All protocols
#	Protocols employing loss detection (with or without pure ACK loss detection)
$	Require a working Spin bit
!	Metric relative to the opposite channel
x2	Same metric for both directions
ppa	Packets-Per-Ack
Q|L	See Q if Upstream loss is significant; L otherwise
E2E	End to end

4.1. Implementation Considerations

By combining the information of the two tables above, it can be deduced that the solutions with 3 bits (i.e., QL or QR + S or D) or 4 bits (i.e., QL or QR + SD) allow having more complete and resilient measurements.

The methodologies described in the previous sections are transport agnostic and can be applied in various situations. The choice of the methods also depends on the specific protocol. For example, QL is a good combination; however, if a protocol does not support, or cannot set, the L bit, QR is the only viable solution.

5. Examples of Application

This document describes several measurement methods, but it is not expected that all methods will be implemented together. For example, only some of the methods described in this document (i.e., sQuare bit and Spin bit) are utilized in [CORE-COAP-PM]. Also, the binding of a delay signal to QUIC is partially described in Section 17.4 of [QUIC-TRANSPORT], which adds only the Spin bit to the first byte of the short packet header, leaving two reserved bits for future use (see Section 17.2.2 of [QUIC-TRANSPORT]).

All signals discussed in this document have been implemented in successful experiments for both QUIC and TCP. The application scenarios considered allow the monitoring of the interconnections inside a data center (Intra-DC), between data centers (Inter-DC), as well as end-to-end large-scale data transfers. For the application of the methods described in this document, it is assumed that the monitored flows follow stable paths and traverse the same measurement points.

The specific implementation details and the choice of the bits used for the experiments with QUIC and TCP are out of scope for this document. A specification defining the specific protocol application is expected to discuss the implementation details depending on which bits will be implemented in the protocol, e.g., [CORE-COAP-PM]. If bits used for specific measurements can also be used for other purposes by a protocol, the specification is expected to address ways for on-path observers to disambiguate the signals or to discuss limitations on the conditions under which the observers can expect a valid signal.

6. Protocol Ossification Considerations

Accurate loss and delay information is not required for the operation of any protocol, though its presence for a sufficient number of flows is important for the operation of networks.

The delay and loss bits are amenable to "greasing" described in [RFC8701] if the protocol designers are not ready to dedicate (and ossify) bits used for loss reporting to this function. The greasing could be accomplished similarly to the latency Spin bit greasing in Section 17.4 of [QUIC-TRANSPORT]. For example, the protocol designers could decide that a fraction of flows should not encode loss and delay information, and instead, the bits would be set to arbitrary values.

Setting any of the bits described in this document to arbitrary values would make the corresponding delay and loss information resemble noise rather than the expected signal for the flow, and the observers would need to be ready to ignore such flows.

7. Security Considerations

The methods described in this document are transport agnostic and potentially applicable to any transport-layer protocol, and especially valuable for encrypted protocols. These methods can be applied to both limited domains and the Internet, depending on the specific protocol application.

Passive loss and delay observations have been a part of the network operations for a long time, so exposing loss and delay information to the network does not add new security concerns for protocols that are currently observable.

In the absence of packet loss, Q and R bits signals do not provide any information that cannot be observed by simply counting packets transiting a network path. In the presence of packet loss, Q and R bits will disclose the loss, but this is information about the environment and not the endpoint state. The L bit signal discloses internal state of the protocol's loss-detection machinery, but this state can often be gleaned by timing packets and observing the congestion controller response.

The measurements described in this document do not imply that new packets injected into the network can cause potential harm to the network itself and to data traffic. The measurements could be harmed by an attacker altering the marking of the packets or injecting artificial traffic. Authentication techniques may be used where appropriate to guard against these traffic attacks.

Hence, loss bits do not provide a viable new mechanism to attack data integrity and secrecy.

The measurement fields introduced in this document are intended to be included in the packets. However, it is worth mentioning that it may be possible to use this information as a covert channel.

This document does not define a specific application, and the described techniques can generally apply to different communication protocols operating in different security environments. A specification defining a specific protocol application is expected to address the respective security considerations and must consider specifics of the protocol and its expected operating environment. For example, security considerations for QUIC, discussed in Section 21 of [QUIC-TRANSPORT] and Section 9 of [QUIC-TLS], consider a possibility of active and passive attackers in the network as well as attacks on specific QUIC mechanisms.

7.1. Optimistic ACK Attack

A defense against an optimistic ACK attack, described in Section 21.4 of [QUIC-TRANSPORT], involves a sender randomly skipping packet numbers to detect a receiver acknowledging packet numbers that have never been received. The Q bit signal may inform the attacker which packet numbers were skipped on purpose and which had been actually lost (and are, therefore, safe for

the attacker to acknowledge). To use the Q bit for this purpose, the attacker must first receive at least an entire Q Block of packets, which renders the attack ineffective against a delay-sensitive congestion controller.

A protocol that is more susceptible to an optimistic ACK attack with the loss signal provided by the Q bit and that uses a loss-based congestion controller should shorten the current Q Block by the number of skipped packets numbers. For example, skipping a single packet number will invert the square signal one outgoing packet sooner.

Similar considerations apply to the R bit, although a shortened Reflection Block along with a matching skip in packet numbers does not necessarily imply a lost packet, since it could be due to a lost packet on the reverse path along with a deliberately skipped packet by the sender.

7.2. Delay Bit with RTT Obfuscation

Theoretically, delay measurements can be used to roughly evaluate the distance of the client from the server (using the RTT) or from any intermediate observer (using the client-observer half-RTT). As described in [RTT-PRIVACY], connection RTT measurements for geolocating endpoints are usually inferior to even the most basic IP geolocation databases. It is the variability within RTT measurements (the jitter) that is most informative, as it can provide insight into the operating environment of the endpoints as well as the state of the networks (queuing delays) used by the connection.

Nevertheless, to further mask the actual RTT of the connection, the Delay bit algorithm can be slightly modified by, for example, delaying the client-side reflection of the delay sample by a fixed, randomly chosen time value. This would lead an intermediate observer to measure a delay greater than the real one.

This Additional Delay should be randomly selected by the client and kept constant for a certain amount of time across multiple connections. This ensures that the client-server jitter remains the same as if no Additional Delay had been inserted. For example, a new Additional Delay value could be generated whenever the client's IP address changes.

Despite the Additional Delay, this Hidden Delay technique still allows an accurate measurement of the RTT components (observer-server) and all the intra-domain measurements used to distribute the delay in the network. Furthermore, unlike the Delay bit, the Hidden Delay bit does not require the use of the client reflection threshold (1 ms by default). Removing this threshold may lead to increasing the number of valid measurements produced by the algorithm.

Note that the Hidden Delay bit does not affect an observer's ability to measure accurate RTT using other means, such as timing packets exchanged during the connection establishment.

8. Privacy Considerations

To minimize unintentional exposure of information, loss bits provide an explicit loss signal -- a preferred way to share information per [RFC8558].

New protocols commonly have specific privacy goals, and loss reporting must ensure that loss information does not compromise those privacy goals. For example, [QUIC-TRANSPORT] allows changing Connection IDs in the middle of a connection to reduce the likelihood of a passive observer linking old and new sub-flows to the same device (see Section 5.1 of [QUIC-TRANSPORT]). A QUIC implementation would need to reset all counters when it changes the destination (IP address or UDP port) or the Connection ID used for outgoing packets. It would also need to avoid incrementing the Unreported Loss counter for loss of packets sent to a different destination or with a different Connection ID.

It is also worth highlighting that, if these techniques are not widely deployed, an endpoint that uses them may be fingerprinted based on their usage. However, since there is no release of user data, the techniques seem unlikely to substantially increase the existing privacy risks.

Furthermore, if there is experimental traffic with these bits set on the network, a network operator could potentially prioritize this marked traffic by placing it in a priority queue. This may result in the delivery of better service, which could potentially mislead an experiment intended to benchmark the network.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

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

[IPPM-METHODS] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

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

[RFC8558] Hardie, T., Ed., "Transport Protocol Path Signals", RFC 8558, DOI 10.17487/RFC8558, April 2019, <https://www.rfc-editor.org/info/rfc8558>.

[TCP] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

10.2. Informative References

[ACCURATE-ECN]

Briscoe, B., Kühlewind, M., and R. Scheffenegger, "More Accurate Explicit Congestion Notification (ECN) Feedback in TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-accurate-ecn-26, 24 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-accurate-ecn-26>.

[AltMark] Fioccola, G., Ed., Cociglio, M., Mirsky, G., Mizrahi, T., and T. Zhou, "Alternate-Marking Method", RFC 9341, DOI 10.17487/RFC9341, December 2022, <https://www.rfc-editor.org/info/rfc9341>.

[ANRW19-PM-QUIC] Bulgarella, F., Cociglio, M., Fioccola, G., Marchetto, G., and R. Sisto, "Performance measurements of QUIC communications", Proceedings of the Applied Networking Research Workshop (ANRW '19), Association for Computing Machinery, DOI 10.1145/3340301.3341127, July 2019, <https://doi.org/10.1145/3340301.3341127>.

[ConEx] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <https://www.rfc-editor.org/info/rfc7713>.

[ConEx-TCP] Kuehlewind, M., Ed. and R. Scheffenegger, "TCP Modifications for Congestion Exposure (ConEx)", RFC 7786, DOI 10.17487/RFC7786, May 2016, <https://www.rfc-editor.org/info/rfc7786>.

[CORE-COAP-PM] Fioccola, G., Zhou, T., Nilo, M., Milan, F., and F. Bulgarella, "Constrained Application Protocol (CoAP) Performance Measurement Option", Work in Progress, Internet-Draft, draft-ietf-core-coap-pm-01, 19 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pm-01>.

[IPPM-SPIN] Trammell, B., Ed., "An Explicit Transport-Layer Signal for Hybrid RTT Measurement", Work in Progress, Internet-Draft, draft-trammell-ippm-spin-00, 9 January 2019, <https://datatracker.ietf.org/doc/html/draft-trammell-ippm-spin-00>.

[IPv6AltMark] Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R. Pang, "IPv6 Application of the Alternate-Marking Method", RFC 9343, DOI 10.17487/RFC9343, December 2022, <https://www.rfc-editor.org/info/rfc9343>.

[QUIC-MANAGEABILITY] Kühlewind, M. and B. Trammell, "Manageability of the QUIC Transport Protocol", RFC 9312, DOI 10.17487/RFC9312, September 2022, <https://www.rfc-editor.org/info/rfc9312>.

[QUIC-SPIN] Trammell, B., Ed., De Vaere, P., Even, R., Fioccola, G., Fossati, T., Ihlar, M., Morton, A., and S. Emile, "Adding Explicit Passive Measurability of Two-Way Latency to the QUIC Transport Protocol", Work in Progress, Internet-Draft, draft-trammell-quic-spin-03, 14 May 2018, <https://datatracker.ietf.org/doc/html/draft-trammell-quic-spin-03>.

[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/info/rfc9001>.

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <https://www.rfc-editor.org/info/rfc8701>.

[RTT-PRIVACY] Trammell, B. and M. Kühlewind, "Revisiting the Privacy Implications of Two-Way Internet Latency Data", Passive and Active Measurement, pp. 73-84, Springer International Publishing, DOI 10.1007/978-3-319-76481-8_6, ISBN 9783319764801, March 2018, <https://doi.org/10.1007/978-3-319-76481-8_6>.

[TRANSPORT-ENCRYPT] Fairhurst, G. and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", RFC 9065, DOI 10.17487/RFC9065, July 2021, <https://www.rfc-editor.org/info/rfc9065>.

[TSVWG-SPIN] Trammell, B., Ed., "A Transport-Independent Explicit Signal for Hybrid RTT Measurement", Work in Progress, Internet-Draft, draft-trammell-tsvwg-spin-00, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-trammell-tsvwg-spin-00>.

[UDP-OPTIONS] Touch, J., "Transport Options for UDP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-options-23, 15 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-udp-options-23>.

[UDP-SURPLUS] Herbert, T., "UDP Surplus Header", Work in Progress, Internet-Draft, draft-herbert-udp-space-hdr-01, 8 July 2019, <https://datatracker.ietf.org/doc/html/draft-herbert-udp-space-hdr-01>.

Acknowledgments

The authors would like to thank the QUIC WG for their contributions, Christian Huitema for implementing Q and L bits in his picoquic stack, and Ike Kunze for providing constructive reviews and helpful suggestions.

Contributors

The following people provided valuable contributions to this document:

Marcus Ihlar
Ericsson
Email: marcus.ihlar@ericsson.com

Jari Arkko
Ericsson
Email: jari.arkko@ericsson.com

Emile Stephan
Orange
Email: emile.stephan@orange.com

Dmitri Tikhonov
LiteSpeed Technologies
Email: dtikhonov@litespeedtech.com

Authors' Addresses

Mauro Cociglio
Telecom Italia - TIM
Via Reiss Romoli, 274
10148 Torino
Italy
Email: mauro.cociglio@outlook.com

Alexandre Ferrieux
Orange Labs
Email: alexandre.ferrieux@orange.com

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
80992 Munich
Germany
Email: giuseppe.fioccola@huawei.com

Igor Lubashev
Akamai Technologies
Email: ilubashe@akamai.com

Fabio Bulgarella
Telecom Italia - TIM
Via Reiss Romoli, 274
10148 Torino
Italy
Email: fabio.bulgarella@guest.telecomitalia.it

Massimo Nilo
Telecom Italia - TIM
Via Reiss Romoli, 274
10148 Torino
Italy
Email: massimo.nilo@telecomitalia.it

Isabelle Hamchaoui
Orange Labs
Email: isabelle.hamchaoui@orange.com

Riccardo Sisto
Politecnico di Torino
Email: riccardo.sisto@polito.it