

Workgroup: IPPM
Internet-Draft:
draft-ietf-ippm-explicit-flow-measurements-02
Published: 13 October 2022
Intended Status: Informational
Expires: 16 April 2023
Authors: M. Cociglio A. Ferrieux
 Telecom Italia - TIM Orange Labs
 G. Fioccola I. Lubashev
 Huawei Technologies Akamai Technologies
 F. Bulgarella M. Nilo
 Telecom Italia - TIM Telecom Italia - TIM
 I. Hamchaoui R. Sisto
 Orange Labs Politecnico di Torino
 Explicit Flow Measurements Techniques

Abstract

This document describes protocol independent methods called Explicit Flow Measurement Techniques that employ few marking bits, inside the header of each packet, for loss and delay measurement. The endpoints, marking the traffic, signal these metrics to intermediate observers allowing them to measure connection performance, and to locate the network segment where impairments happen. Different alternatives are considered within this document. These signaling methods apply to all protocols but they are especially valuable when applied to protocols that encrypt transport header and do not allow traditional methods for delay and loss detection.

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 16 April 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

- [1. Introduction](#)
- [2. Latency Bits](#)
 - [2.1. Spin Bit](#)
 - [2.2. Delay Bit](#)
 - [2.2.1. Generation Phase](#)
 - [2.2.2. Reflection Phase](#)
 - [2.2.3. T Max Selection](#)
 - [2.2.4. Delay Measurement Using Delay Bit](#)
 - [2.2.4.1. RTT Measurement](#)
 - [2.2.4.2. Half-RTT Measurement](#)
 - [2.2.4.3. Intra-Domain RTT Measurement](#)
 - [2.2.5. Observer's Algorithm](#)
 - [2.2.6. Two Bits Delay Measurement: Spin Bit + Delay Bit](#)
- [3. Loss Bits](#)
 - [3.1. T Bit -- Round Trip Loss Bit](#)
 - [3.1.1. Round Trip Loss](#)
 - [3.1.2. Setting the Round Trip Loss Bit on Outgoing Packets](#)
 - [3.1.3. Observer's Logic for Round Trip Loss Signal](#)
 - [3.1.4. Loss Coverage and Signal Timing](#)
 - [3.2. Q Bit -- sQuare Bit](#)
 - [3.2.1. Q Block Length Selection](#)
 - [3.2.2. Upstream Loss](#)
 - [3.2.3. Identifying Q Block Boundaries](#)
 - [3.2.3.1. Improved Resilience to Burst Losses](#)
 - [3.3. L Bit -- Loss Event Bit](#)
 - [3.3.1. End-To-End Loss](#)
 - [3.3.1.1. Loss Profile Characterization](#)
 - [3.3.2. L+Q Bits -- Loss Measurement Using L and Q Bits](#)
 - [3.3.2.1. Correlating End-to-End and Upstream Loss](#)
 - [3.3.2.2. Downstream Loss](#)
 - [3.3.2.3. Observer Loss](#)

- [3.4. R Bit -- Reflection Square Bit](#)
 - [3.4.1. Enhancement of R Block Length Computation](#)
 - [3.4.2. Improved Resilience to Packet Reordering](#)
 - [3.4.2.1. Improved Resilience to Burst Losses](#)
 - [3.4.3. R+Q Bits -- Loss Measurement Using R and Q Bits](#)
 - [3.4.3.1. Three-Quarters Connection Loss](#)
 - [3.4.3.2. End-To-End Loss in the Opposite Direction](#)
 - [3.4.3.3. Half Round-Trip Loss](#)
 - [3.4.3.4. Downstream Loss](#)
- [3.5. E Bit -- ECN-Echo Event Bit](#)
 - [3.5.1. Setting the ECN-Echo Event Bit on Outgoing Packets](#)
 - [3.5.2. Using E Bit for Passive ECN-Reported Congestion Measurement](#)
- [4. Summary of Delay and Loss Marking Methods](#)
- [5. Protocol Ossification Considerations](#)
- [6. Examples of Application](#)
 - [6.1. QUIC](#)
 - [6.2. TCP](#)
- [7. Security Considerations](#)
 - [7.1. Optimistic ACK Attack](#)
 - [7.2. Delay Bit with RTT Obfuscation](#)
- [8. Privacy Considerations](#)
- [9. IANA Considerations](#)
- [10. Contributors](#)
- [11. Acknowledgements](#)
- [12. References](#)
 - [12.1. Normative References](#)
 - [12.2. Informative References](#)
- [Authors' Addresses](#)

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 crippling end-to-end throughput issues. To this effect, in a TCP-dominated world, network operators have been heavily relying on information present in the clear in TCP headers: sequence and acknowledgement numbers and SACKs when enabled (see [[RFC8517](#)]). These allow for quantitative estimation of packet loss and delay by passive on-path observation. Additionally, the problem can be quickly identified in the network path by moving the passive observer around.

With encrypted protocols, the equivalent transport headers are encrypted and passive packet loss and delay observations are not possible, as described in [[RFC9065](#)].

Measuring TCP loss and delay between similar endpoints cannot be relied upon to evaluate encrypted protocol loss and delay. Different protocols could be routed by the network differently, and the fraction of Internet traffic delivered using protocols other than TCP is increasing every year. It is imperative to measure packet loss and delay experienced by encrypted protocol users directly.

This document defines Explicit Flow Measurement Techniques. These hybrid measurement path signals (see [[IPM-Methods](#)]) are to be embedded into a transport layer protocol and are explicitly intended for exposing RTT and loss rate information to on-path measurement devices. They are designed to facilitate network operations and management and are "beneficial" for maintaining the quality of service (see [[RFC9065](#)]). These measurement mechanisms are applicable to any transport-layer protocol, and, as an example, the document describes QUIC and TCP bindings.

The Explicit Flow Measurement Techniques described in this document can be used alone or in combination with other Explicit Flow Measurement Techniques. Each technique uses a small number of bits and exposes a specific measurement.

Following the recommendation in [[RFC8558](#)] of making path signals explicit, this document proposes adding a small number of dedicated measurement bits to the clear portion of the protocol headers. These bits can be added to an unencrypted portion of a header belonging to any protocol layer, e.g. IP (see [[IP](#)]) and IPv6 (see [[IPv6](#)]) headers or extensions, such as [[IPv6AltMark](#)], UDP surplus space (see [[UDP-OPTIONS](#)] and [[UDP-SURPLUS](#)]), reserved bits in a QUIC v1 header, as already done with the latency Spin bit (see [[QUIC-TRANSPORT](#)]).

The measurements are not designed for use in automated control of the network in environments where signal bits are set by untrusted hosts. Instead, the signal is to be used for troubleshooting individual flows as well as for monitoring the network by aggregating information from multiple flows and raising operator alarms if aggregate statistics indicate a potential problem.

The Spin bit, Delay bit and loss bits explained in this document are inspired by [[AltMark](#)], [[SPIN-BIT](#)], [[I-D.trammell-tsvwg-spin](#)] and [[I-D.trammell-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.

[[QUIC-TRANSPORT](#)] 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. [[SPIN-BIT](#)] discusses an additional two-bit Valid Edge Counter (VEC) to compensate for loss and reordering of the Spin bit and increase fidelity of the signal in less than ideal network conditions.

This document introduces a stand-alone 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, please see [[SPIN-BIT](#)].

The Spin bit is an Alternate-Marking [[AltMark](#)] generated signal, 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 period" is a set of packets with the same Spin bit value sent during one RTT time interval. A "spin period value" is the value of the Spin bit shared by all packets in a spin 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;

- *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 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 periods and therefore wrong RTT measurements;
- *application-limited senders cause the Spin bit to measure the application delays instead of network delays.

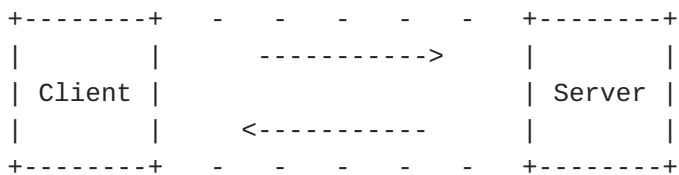
Unlike the Spin bit, which is set in every packet transmitted on the network, the Delay bit is set only once per round trip.

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 "delay sample".

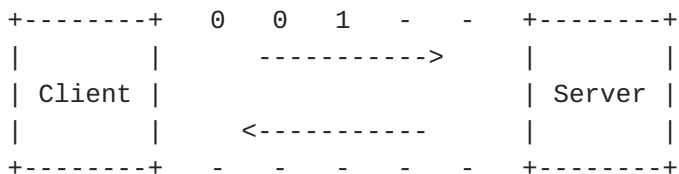
An observer placed at an intermediate point, observing a single direction of traffic, tracking the delay sample and the relative timestamp, can measure the round trip delay of the connection.

The delay sample lifetime is comprised of 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.

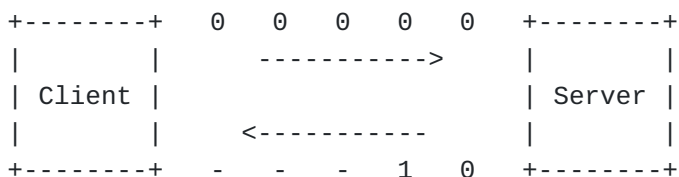
The next figure describes the elementary Delay bit mechanism.



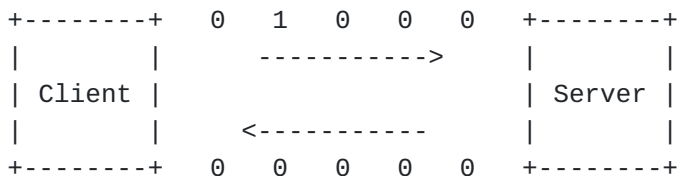
(a) No traffic at beginning.



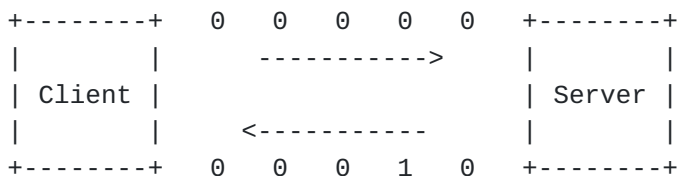
(b) The Client starts sending data and sets the first packet as Delay Sample.



(c) The Server starts sending data and reflects the Delay Sample.



(d) The Client reflects the Delay Sample.



(e) The Server reflects the Delay Sample and so on.

Figure 1: Delay bit mechanism

2.2.1. Generation Phase

Only 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 client and server continuously, for the entire duration of the connection. 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 (1ms 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 on the endpoints. Hence, the maximum estimation error would amount to twice the threshold (e.g. 2ms) 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 select the `T_Max` value so that both 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 expectable 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, 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.

`T_Max_c` could be computed as two times the measured RTT plus a fixed amount of time (100ms) to prevent low `T_Max` values in case of very small RTTs. The resulting formula is: $T_Max_c = 2RTT + 100ms$. If `T_Max_c` is greater than `T_Max_p` then `T_Max_c` is forced to `T_Max_p` value.

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`.

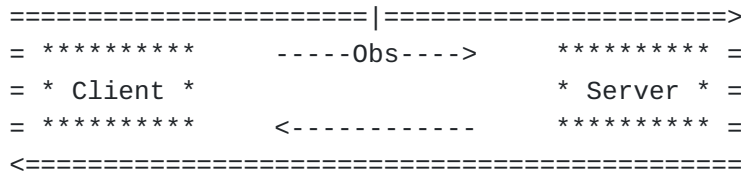
2.2.4. Delay Measurement Using Delay Bit

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.

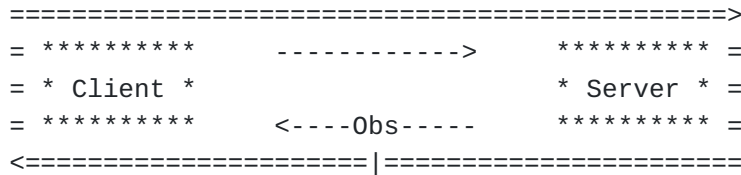
2.2.4.1. RTT Measurement

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} .



(a) client-server RTT



(b) server-client RTT

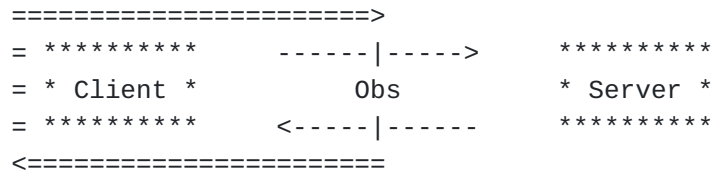
Figure 2: Round-trip time (both direction)

2.2.4.2. Half-RTT Measurement

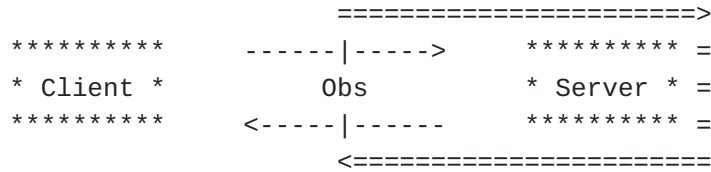
An observer that is able to observe both forward and return traffic directions can use the delay samples to measure "upstream" and "downstream" RTT components, also known as the half-RTT measurements. It does this by measuring the time between a delay sample observed in one direction and the delay sample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



(a) client-observer half-RTT

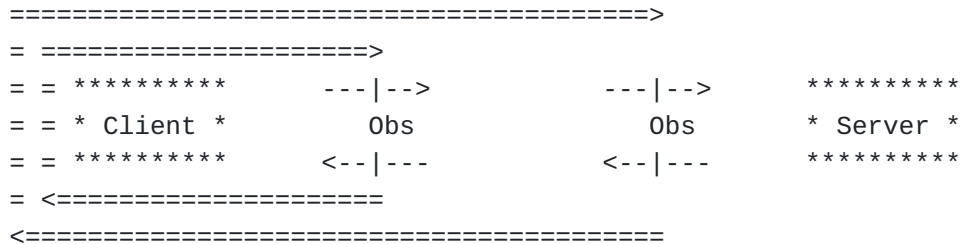


(b) observer-server half-RTT

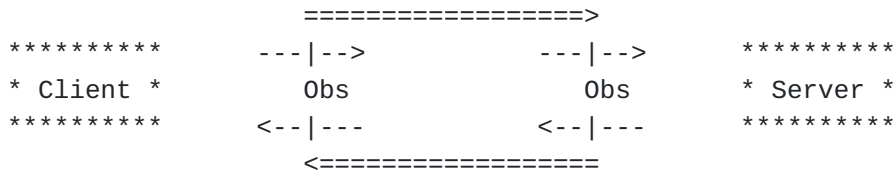
Figure 3: Half Round-trip time (both direction)

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 ingress and egress of the network to be measured. Intra-domain RTT is difference between the two computed upstream (or downstream) RTT components.



(a) client-observer RTT components (half-RTTs)



(b) the intra-domain RTT resulting from the subtraction of the above 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 time at which the last delay sample has been observed.

A unidirectional observer, 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), 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.

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

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;

- *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 signal" 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 signal" bit is used in combination with Q bit to measure end-to-end loss. See [Section 3.1](#).

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 separately identifiable flow (each sub-flow for multipath connections).

Since loss is reported independently for each flow, all bits (except for 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, generates and consequently transmits a first train of packets, by setting the T bit to 1;
- *the server, upon receiving each packet included in the first train, reflects to the client a respective second train of packets of the same size as the first train received, by setting the T bit to 1;
- *the client, upon receiving each packet included in the second train, reflects to the server a respective third train of packets of the same size as the second train received, by setting the T bit to 1;
- *the server, upon receiving each packet included in the third train, finally reflects to the client a respective fourth train of packets of the same size as the third train received, by setting the T bit to 1.

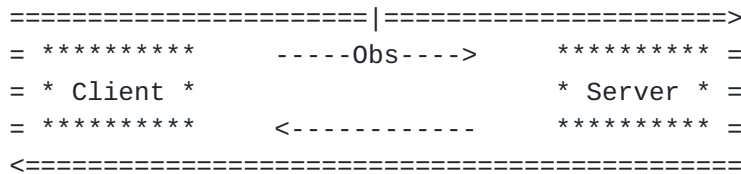
Packets belonging to the first round trip (first and second train) represent the Generation Phase, while those belonging to the second round trip (third and fourth train) represent the Reflection Phase.

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.

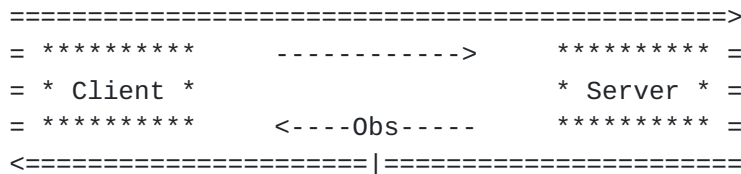
Since packet rates in two directions may be different, the number of marked packets in the train is determined by the direction with the lowest packet rate. See [Section 3.1.2](#) for details on packet generation.

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.



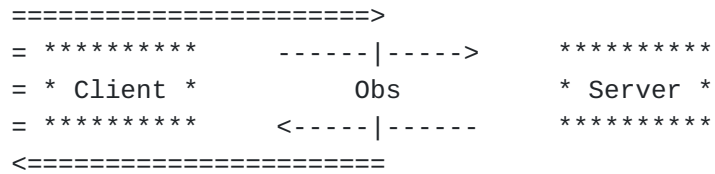
(a) client-server RTPL



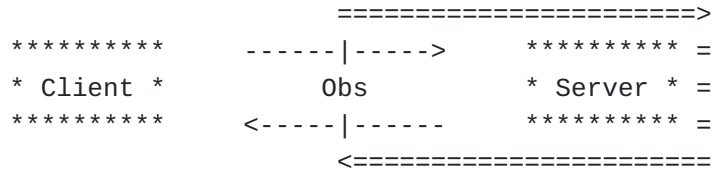
(b) server-client RTPL

Figure 5: Round-trip packet loss (both direction)

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

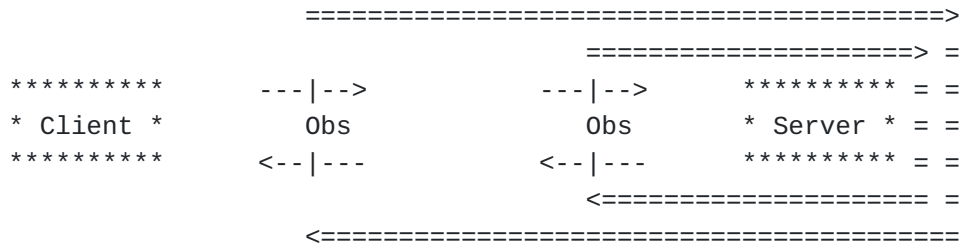


(a) client-observer half-RTPL

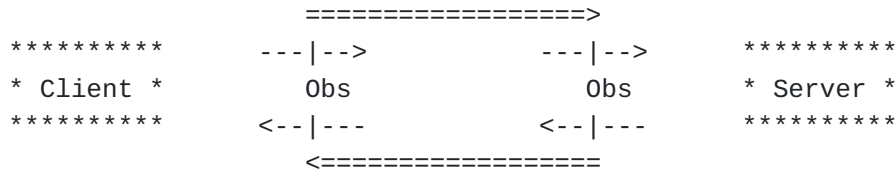


(b) observer-server half-RTPL

Figure 6: Half Round-trip packet loss (both direction)



(a) observer-server RTPL components (half-RTPLs)



(b) the intra-domain RTPL resulting from the subtraction of the above RTPL components

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 separator for the on-path observer.

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 reached zero. The "generation token" method from the generation phase is used during this phase as well. At the end of the first spin-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. The most conservative cap value is 1.

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 periods) for the generation phase is a tradeoff 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 generation, ~2 reflection, 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 just ~1/4 of packets in the slower direction.

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):

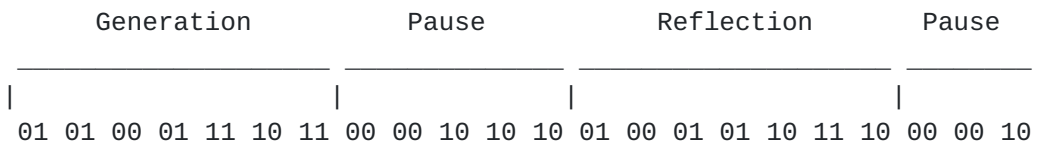


Figure 8: Round Trip Loss signal example

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 two 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 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, $\text{lowPacketRate}/(3*\text{highPacketRate})$ 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 $1/(3*\text{packetsPerAck})$, where the value of `packetsPerAck` can vary by protocol, by implementation, and by network conditions.

3.2. Q Bit -- sSquare Bit

The sSquare 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 allows to mark consecutive blocks of packets with different colors.

[[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. The method based on fixed timer can measure packet loss on a network segment by cooperating and synchronized observers on both ends of the segment comparing packets 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 Network Management System (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 sSquare 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 implements 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 5](#)).

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.

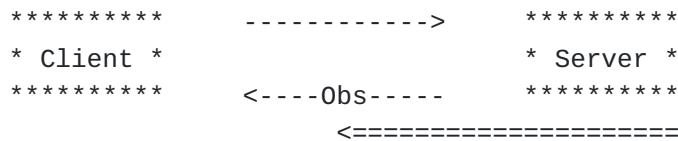
3.2.2. Upstream Loss

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](#).



(a) in client-server channel (uloss_up)



(b) in server-client channel (uloss_down)

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 Q measurements accuracy. Generally, burst losses can be absorbed and correctly measured if smaller than the established Q Block length. If entire Q Block length of packets get 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 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 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 SCTP (losses of pure ACKs are not detected) and is not true of protocols such as UDP and IP/IPv6.

The Unreported Loss counter is initialized to 0, and 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, but it should not become negative due to the rescission.

This loss signaling is similar to loss signaling in [[ConEx](#)], except the Loss Event bit is reporting the exact number of lost packets, whereas Echo Loss bit 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 Unreported Loss counter by the fraction of the packet data lost (so 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 value of 0 and 1 for a given flow. The end-to-end loss rate 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 loss profile, since the distribution of observed packets with L bit set to 1 roughly corresponds to the distribution of packets lost between 1 RTT and 1 RTO before (see [Section 3.3.2.1](#)). Hence, observing random single instances of L bit set to 1 indicates random single packet loss, while observing blocks of packets with 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 acknowledgements) and 1 RTO (loss declared due to a timeout) relative to the upstream loss.

The flow RTT can sometimes be estimated by timing 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 use 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 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 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 signal 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;

- *if, before transmission of the block is terminated, the reception of at least one further Q Block is completed, 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" (R Block).

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

The parameter $\text{avg}(p)$, the average number of packets in a marking period, is computed based on all the Q Blocks received since the beginning of the current R Block.

The transmission of an R 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.

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 R 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 R 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 R Block.

This can be achieved introducing the new r_avg parameter in the computation of M . The new formula is $M_r=\text{avg}(p)+r_avg$; $M=\text{round}(M_r)$; $r_avg=M_r-M$ 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 using methods described in [Section 3.2.3](#).

This can be achieved by updating the size of the current R 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 R measurements accuracy similarly to how they affect Q measurements accuracy. Therefore, recommendations in section [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 sSquare 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):

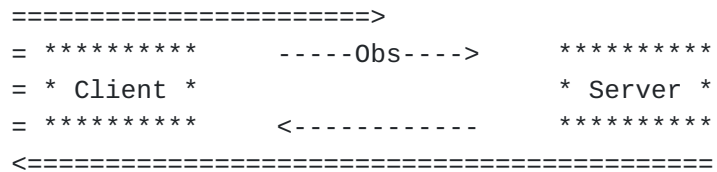
- *client-observer half round-trip loss: the loss between the client and the observation point in both directions

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

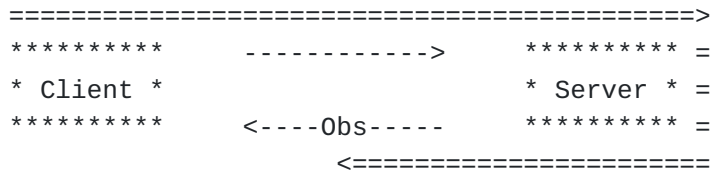
*downstream loss: the loss between the observation point and the receiver (applicable to both directions)

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. Knowing the value of N, an on-path observer can estimate the amount of loss 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 ($tqloss = 1 - avg(t)/N$).



(a) in client-server channel (tqloss_up)



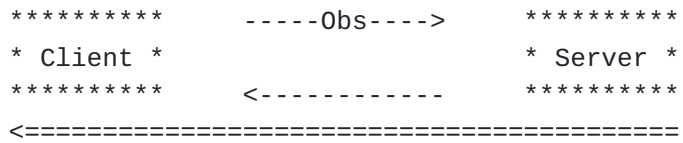
(b) in server-client channel (tqloss_down)

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, $eloss_unobserved = (tqloss - uloss) / (1 - uloss)$.



(a) in client-server channel (eloss_down)

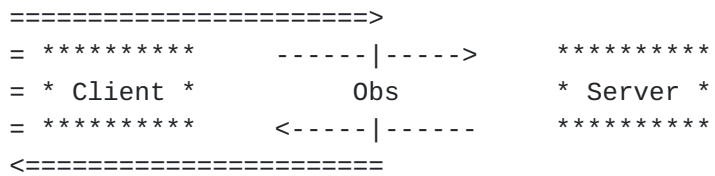


(b) in server-client channel (eloss_up)

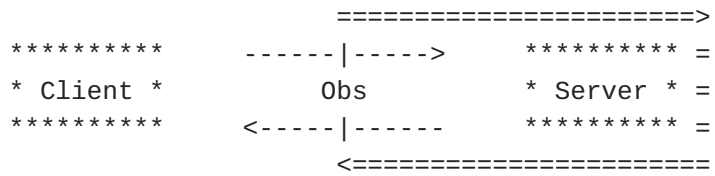
Figure 11: End-To-End loss in the opposite direction

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, $hrtloss=(tqloss_opposite-uloss)/(1-uloss)$.



(a) client-observer half round-trip loss (hrtloss_co)



(b) observer-server half round-trip loss (hrtloss_os)

Figure 12: Half Round-trip loss (both direction)

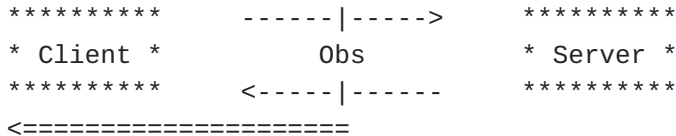
3.4.3.4. Downstream Loss

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 using "half round-trip" loss and upstream loss in the opposite direction.

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



(a) in client-server channel (dloss_up)



(b) in server-client channel (dloss_down)

Figure 13: Downstream loss

3.5. E Bit -- ECN-Echo Event Bit

While the primary focus of this draft 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 CE markings, according to the ECN-Echo feedback from the receiver.

This ECN-Echo signaling is similar to ECN signaling in [ConEx]. 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].

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

A network observer can count packets with 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 a E bit equal to 1.

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

4. Summary of Delay and Loss Marking Methods

This section summarizes the marking methods described in this draft.

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	low	very
			Half RTT		high
D: Delay Bit	1	RTT	x2	high	medium
			Half RTT		
SD: Spin Bit & Delay Bit *	2	RTT	x2	high	very
			Half RTT		high

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

Figure 14: Delay Comparison

For the Loss measurement, each row in the table of [Figure 15](#) represents a loss marking method. For each method the table specifies the number of bits required in the header, the available metrics using an unidirectional or bidirectional observer, applicable protocols, measurement fidelity and delay.

Method	B	Available Loss Metrics	P	Measurement Aspects
	i	UniDir Observer	r	Fidelity
	t	BiDir Observer	t	Delay
	s		o	
T: Round Trip Loss Bit	\$	RT	x2	Rate by ~6 RTT
	1	Half RT	*	sampling +-----
				1/3 to 1/(3*ppa) of
				pkts over 2 RTT
Q: sQuare Bit	1	Upstream	x2	Rate over N pkts
				N pkts (e.g. 64)
				(e.g. 64)
L: Loss Event Bit	1	E2E	x2	Loss shape Min: RTT
				(and rate) Max: RT0
QL: sQuare + Loss Ev. Bits	2	Upstream Downstream E2E	x2 x2 x2	-> see Q Up: see Q Others: see L
				-> see Q L
				-> see L
QR: sQuare + Ref. Sq. Bits	2	Upstream 3/4 RT !E2E	x2 x2 E2E	Rate over N*ppa pkts (see Q bit for N) Up: see Q Others: N*ppa pk (see Q for N)
				Downstream
				Half RT

- * 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

Figure 15: Loss Comparison

5. Protocol Ossification Considerations

Accurate loss and delay information is not critical to 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 [QUIC-TRANSPORT]. Namely, implementations could decide that a fraction of flows should not encode loss and delay information and, instead, the bits would be set to arbitrary values. The observers would need to be ready to ignore flows with delay and loss information more resembling noise than the expected signal.

6. Examples of Application

6.1. QUIC

The binding of a delay signal to QUIC is partially described in [QUIC-TRANSPORT], which adds the Spin bit to the first byte of the short packet header, leaving two reserved bits for future use.

To implement the additional signals discussed in this document, the first byte of the short packet header can be modified as follows:

*the Delay bit (D) can be placed in the first reserved bit (i.e. the fourth most significant bit $0x10$) while the round trip loss bit (T) in the second reserved bit (i.e. the fifth most significant bit $0x08$); the proposed scheme is:

```
 0 1 2 3 4 5 6 7
+--+--+--+--+--+
|0|1|S|D|T|K|P|P|
+--+--+--+--+--+
```

Figure 16: Scheme 1

*alternatively, a two bits loss signal (QL or QR) can be placed in both reserved bits; the proposed schemes, in this case, are:

```
 0 1 2 3 4 5 6 7
+--+--+--+--+--+
|0|1|S|Q|L|K|P|P|
+--+--+--+--+--+
```

Figure 17: Scheme 2A

```

0 1 2 3 4 5 6 7
+--+--+--+--+--+
|0|1|S|Q|R|K|P|P|
+--+--+--+--+--+

```

Figure 18: Scheme 2B

A further option would be to substitute the Spin bit with the Delay bit, leaving the two reserved bits for loss detection. The proposed schemes are:

```

0 1 2 3 4 5 6 7
+--+--+--+--+--+
|0|1|D|Q|L|K|P|P|
+--+--+--+--+--+

```

Figure 19: Scheme 3A

```

0 1 2 3 4 5 6 7
+--+--+--+--+--+
|0|1|D|Q|R|K|P|P|
+--+--+--+--+--+

```

Figure 20: Scheme 3B

6.2. TCP

The signals can be added to TCP by defining bit 4 of byte 13 of the TCP header to carry the Spin bit or the Delay bit, and possibly bits 5 and 6 to carry additional information, like the Delay bit and the round-trip loss bit (DT), or a two bits loss signal (QL or QR).

7. Security Considerations

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 gleamed by timing packets and observing congestion controller response.

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

The described techniques can generally apply to different communication protocols operating in different security environments. An implementation of these techniques for a particular protocol must consider specifics of the protocol and its expected operating environment. For example, security considerations for QUIC, discussed in [[QUIC-TRANSPORT](#)] and [[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 [[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 Q bit and 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 R 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 [[SPIN-RTT](#)], 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 (1ms by default). Removing this threshold may lead to increasing the number of valid measurements produced by the algorithm.

Note that 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. 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 Unreported Loss counter for loss of packets sent to a different destination or with a different Connection ID.

9. IANA Considerations

This document makes no request of IANA.

10. Contributors

The following people provided valuable contributions to this document:

*Marcus Ihlar, Ericsson, marcus.ihlar@ericsson.com

*Jari Arkko, Ericsson, jari.arkko@ericsson.com

*Emile Stephan, Orange, emile.stephan@orange.com

*Dmitri Tikhonov, LiteSpeed Technologies,
dtikhonov@litespeedtech.com

11. Acknowledgements

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.

12. References

12.1. Normative References

- [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>>.
- [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>>.
- [IP] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [IPM-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>>.

[IPv6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

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

[TCP] Postel, J., "Transmission Control Protocol", RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.

12.2. Informative References

[ACCURATE] Briscoe, B., Kühlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-accurate-ecn-20, 25 July 2022, <<https://www.ietf.org/archive/id/draft-ietf-tcpm-accurate-ecn-20.txt>>.

[AltMark] Fioccola, G., Cociglio, M., Mirsky, G., Mizrahi, T., and T. Zhou, "Alternate-Marking Method", Work in Progress, Internet-Draft, draft-ietf-ippm-rfc8321bis-03, 25 July 2022, <<https://www.ietf.org/archive/id/draft-ietf-ippm-rfc8321bis-03.txt>>.

[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, DOI 10.1145/3340301.3341127, July 2019, <<https://doi.org/10.1145/3340301.3341127>>.

[I-D.trammell-ippm-spin] Trammell, B., "An Explicit Transport-Layer Signal for Hybrid RTT Measurement", Work in Progress, Internet-Draft, draft-trammell-ippm-spin-00, 9 January 2019, <<https://www.ietf.org/archive/id/draft-trammell-ippm-spin-00.txt>>.

[I-D.trammell-tsvwg-spin] Trammell, B., "A Transport-Independent Explicit Signal for Hybrid RTT Measurement", Work in Progress, Internet-Draft, draft-trammell-tsvwg-spin-00, 2

July 2018, <<https://www.ietf.org/archive/id/draft-trammell-tsvwg-spin-00.txt>>.

[IPv6AltMark] Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R. Pang, "IPv6 Application of the Alternate Marking Method", Work in Progress, Internet-Draft, draft-ietf-6man-ipv6-alt-mark-17, 27 September 2022, <<https://www.ietf.org/archive/id/draft-ietf-6man-ipv6-alt-mark-17.txt>>.

[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>>.

[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>>.

[RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.

[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>>.

[RFC9065] 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>>.

[SPIN-BIT] 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>>.

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

[UDP-OPTIONS] Touch, J. D., "Transport Options for UDP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-options-18, 26 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-tsvwg-udp-options-18.txt>>.

[UDP-SURPLUS]

Herbert, T., "UDP Surplus Header", Work in Progress, Internet-Draft, draft-herbert-udp-space-hdr-01, 8 July 2019, <<https://www.ietf.org/archive/id/draft-herbert-udp-space-hdr-01.txt>>.

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