New Spin bit enabled measurements with one or two more bits

draft-cfb-tsvwg-spinbit-new-measurements-00

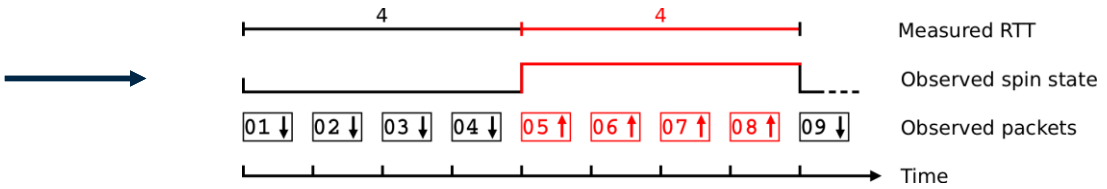
Singapore, November 2019, IETF 106 – TSVWG

Mauro COCIGLIO (Telecom Italia - TIM)
Fabio BULGARELLA (Politecnico di Torino)
Giuseppe FIOCCOLA (Huawei)
Riccardo SISTO (Politecnico di Torino)

How Spin Bit works (draft-trammell-tsvwg-spin)

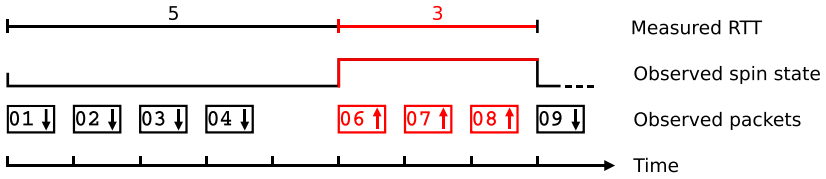
- The spin bit is a simple enhancement of a client-server protocol that causes one bit in the header to ‘spin’, generating one edge (a transition from 0 to 1 or from 1 to 0) once per end-to-end RTT.
- Any device on path can then measure the time (on its local clock) between these edges to generate one RTT sample per RTT for each flow in the general case.
- **SERVER REFLECTS**: when a packet should be sent, it sets the spin bit to the spin bit on the last packet received from the client.
- **CLIENT INVERTS**: when a packet should be sent, it sets the spin bit to the inverse of the spin bit on the last packet received from the server.

Example

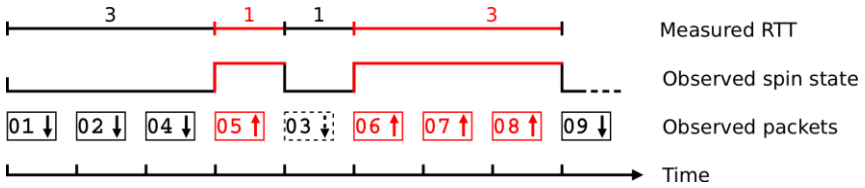


3 Spin Bit limitations

1. Packet loss will tend to cause wrong estimates of RTT due to period width changes.



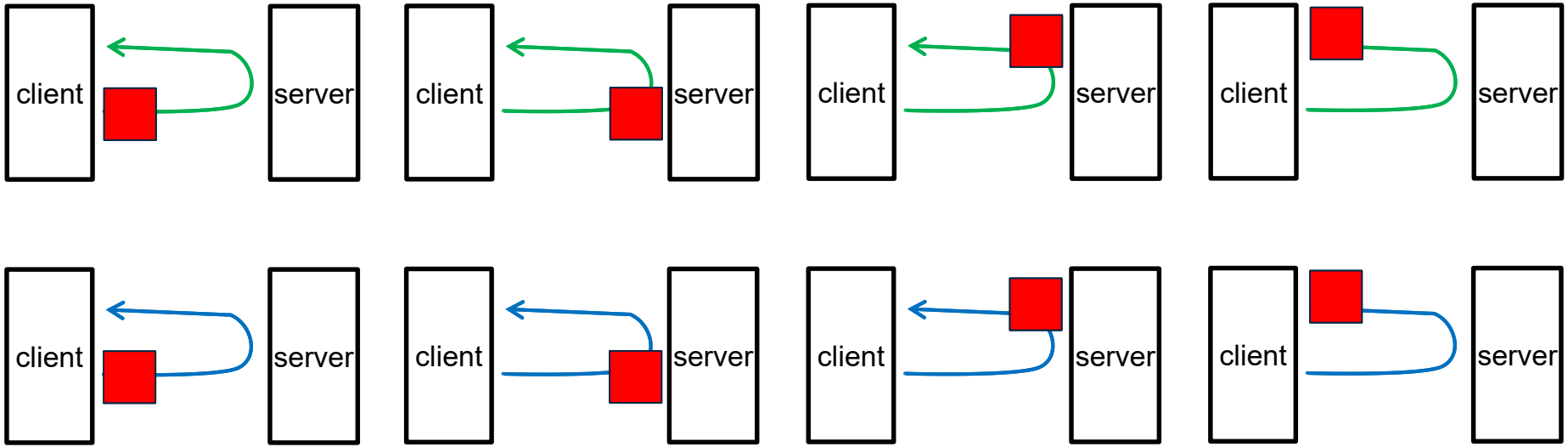
2. Reordering of a spin edge will cause drastic underestimates of RTT since it will cause multiple edges to be observed per RTT. So we need an extra instrument to correctly recognize periods, eluding overlapping.



3. “Holes” in the traffic flow can introduce delay in the edge reflection.

The Delay Bit

- The idea is to have a single packet, with a second marked bit, called «delay bit», that bounces between client and server. This packet is also called ‘’Delay Sample’’.
- A passive observer, placed on whatever direction, can compute the difference in time between two consecutive delay sample determining the RTT of the connection.



Delay Sample red, Spin Bit green or blue

Delay Bit: how the Delay sample works

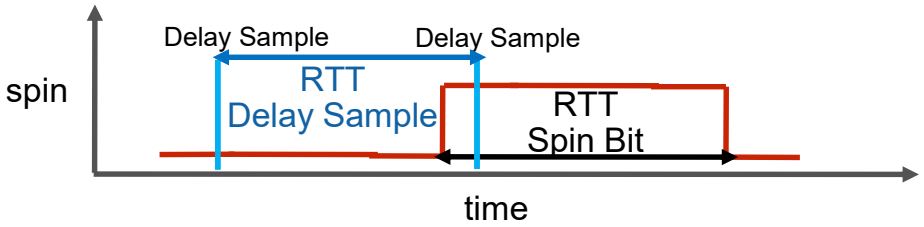
- **Generation:** when the connection starts, the client set the delay bit of the first packet to 1
- **Reflection:** both endpoints reflect an incoming delay sample to the first outgoing packet
  - If reflection is delayed for more than 1ms (due to lack of traffic), reflection is aborted
- **Client side control:** if a spin-bit period ends without a delay sample
  - the **recovery process** is triggered:
    - the client waits an **empty period** in which no delay sample is generated;
    - then, it regenerate the delay sample marking the first packet of the following spin-bit period.

- The empty period is needed to signal to possible observer that there was an issue and a new delay measurement session is starting.

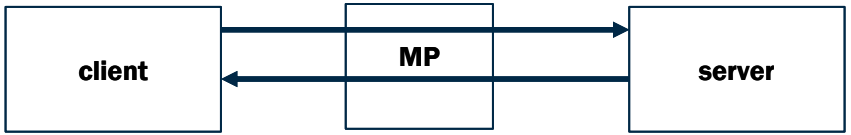
How Delay Sample improves Spin Bit mechanism

Key Goal: stabilize RTT measurements influenced by packet loss, reordering and holes:

- **Packet Loss** → already solved by Delay Sample working principles (single sample for period, empty period when it is lost).
- **Packet Reordering** → can be solved introducing the **waiting interval** into the observer.
  - It is implemented using an interval added after a Spin Bit change. At the end of this waiting period it's possible to change the spin value again. We use a timer (e.g. 5 ms) to implement this waiting time (that is the minimum measurable RTT).
  - This gives us the possibility to correctly identify periods and the related Delay Sample, as well as empty periods used by client to inform observer that there was a loss or a delay so the sample was discarded.
- **Traffic holes** → delayed delay samples are not reflected by the endpoint

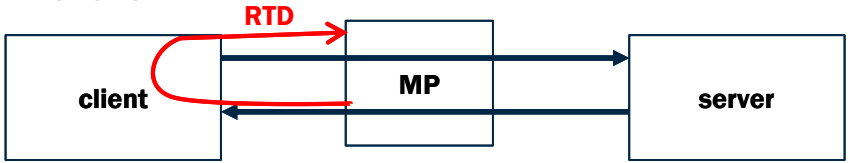


2Point Round-Trip Delay (1 observer)

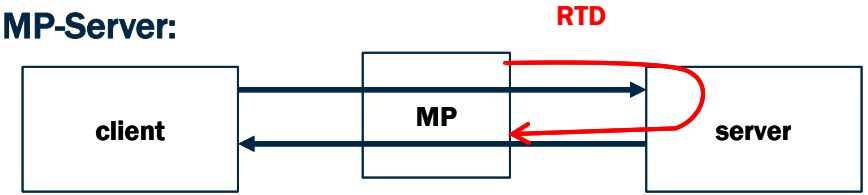


MP= Measurement Point (Observer)

- **RTD MP-Client:**

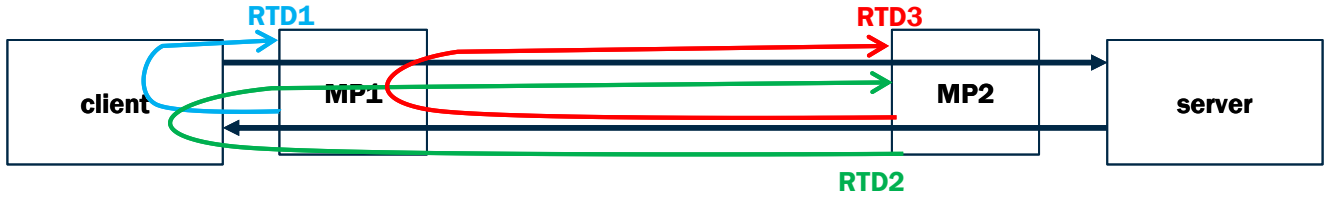


- **RTD MP-Server:**



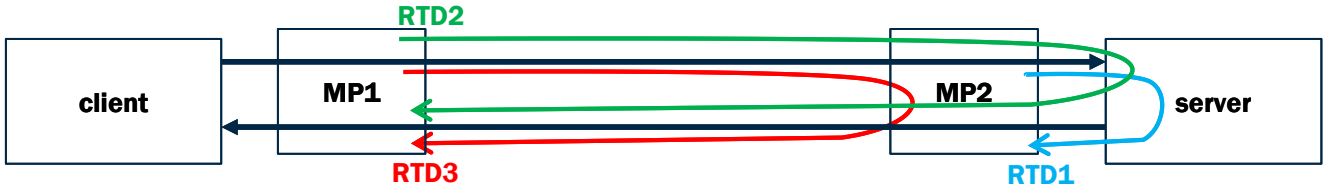
2Point Round-Trip Delay (2 observers)

- RTD MP2-MP1:



MP2-MP1 Round-Trip: RTD2 - RTD1 = RTD3

- RTD MP1-MP2:



MP1-MP2 Round-Trip: RTD2 - RTD1 = RTD3

RoundTrip Packet Loss

- **A new performance metric**, the **RoundTrip Packet Loss**
  - Measured on production traffic between Client and Server.

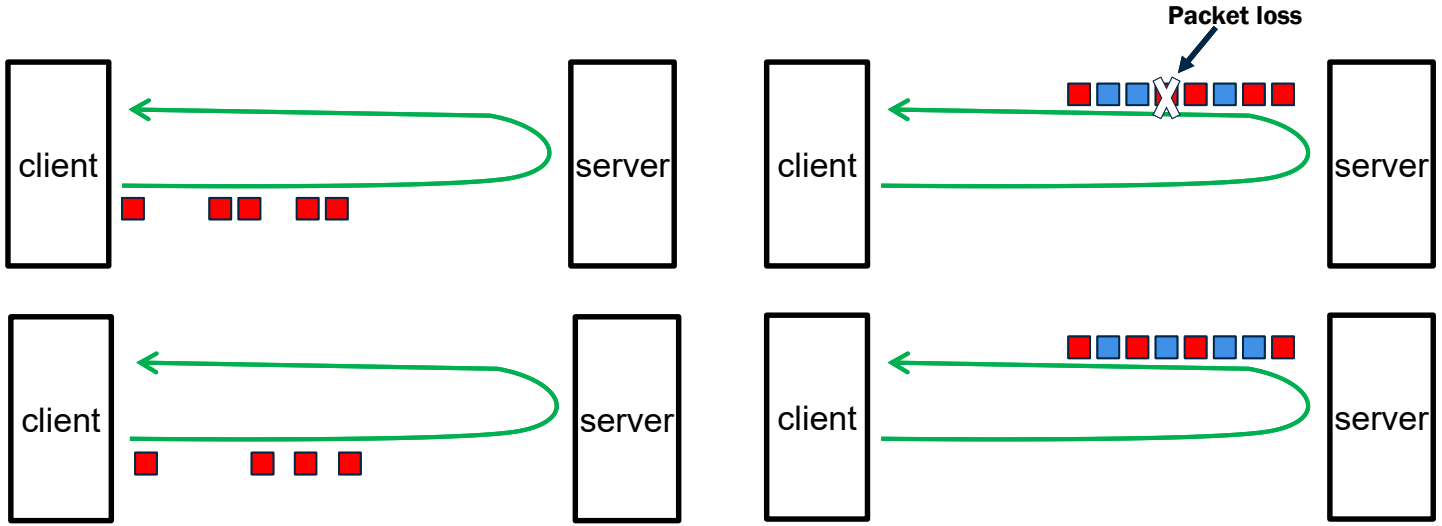
- **How it works**:
  - The Client marks «a train» of production packets and these marked packets «bounces» between Client and Server to complete 2 rounds.
  - Client and Server «reflects» marked packets by marking production packets flowing in the opposite direction.
  - An Observer counts the marked packets during the 2 rounds and compares numbers to find losses.

- **The main issue**: Uplink and Downlink usually have different packet rates.
  - QUESTION: How many packets to mark to avoid marked packets congestion on the slowest traffic direction?
  - ANSWER: the number of packets that transit, in the marking period, on the slowest direction (it's implemented using a token system).

Roundtrip Packet Loss: how it works (1)

- The Client generate a train of market packets (using the Packet Loss bit)
- The Server «reflects» these packets (marking production packets flowing in the opposite direction).
  The Server inserts some not marked packets if downlink rate is greater than uplink rate.
- The Client reflects the marked packets.
- The Server again reflects the marked packets
- The Client generate a new train of market packets and so on.

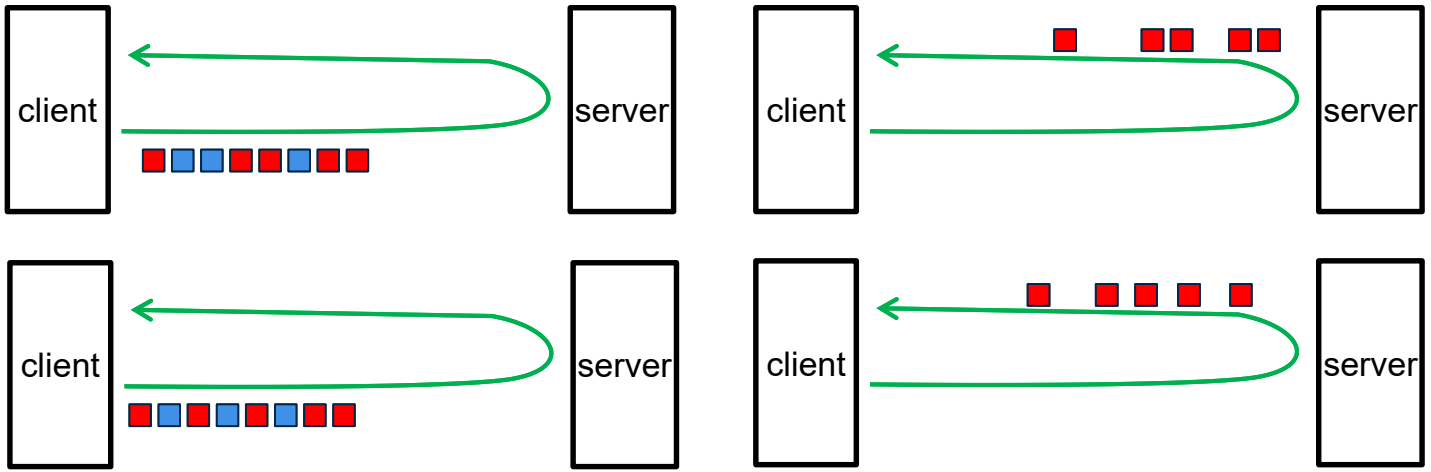
Download flow has more packets



Marked packets: red,
Not Market packets: blue

Roundtrip Packet Loss: how it works (2)

- When upload flow has more packets than download flow we use a token system to maintain the same marked packets rate on both directions (upload and download):

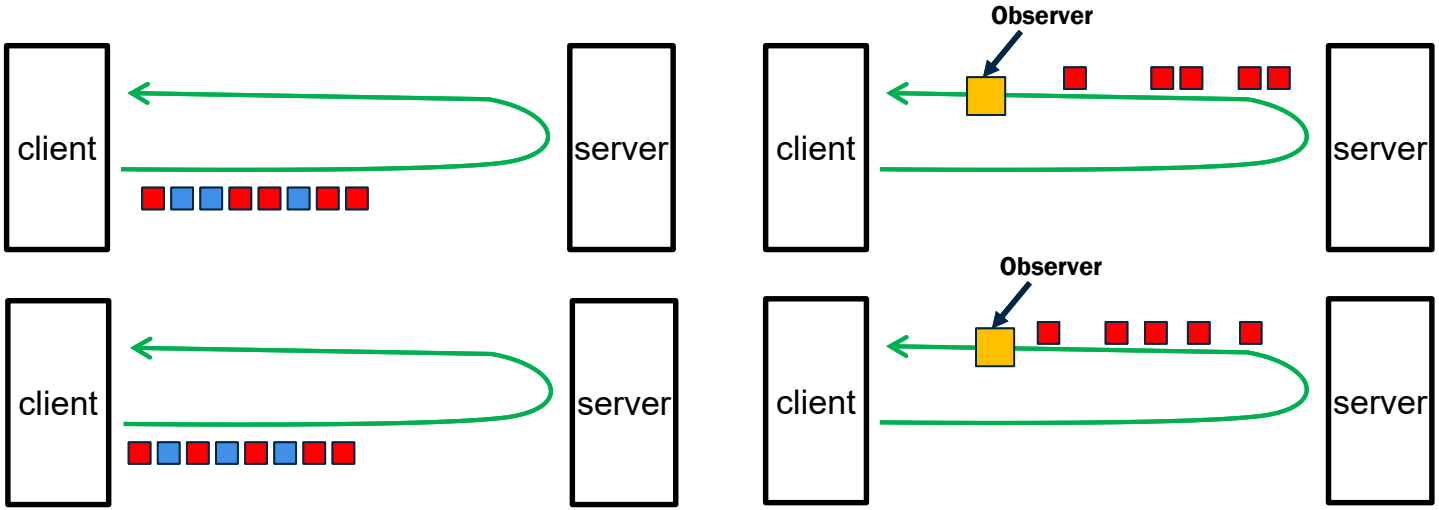


Upload flow has more packets

Marked packets: red, Not Marked packets: blue

Roundtrip Packet Loss: the Observer

- The Observer in the middle (upstream or downstream) sees the packet train twice and so it calculates the «observer roundtrip packet loss» that, statistically, will be equal to the «end-to-end roundtrip packet loss».



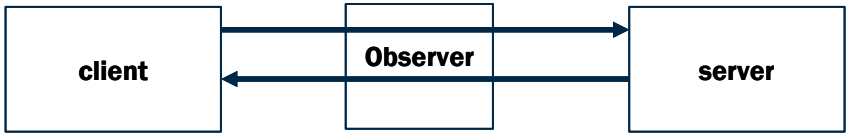
Upload flow has more packets

Marked packets: red, Not Marked packets: blue

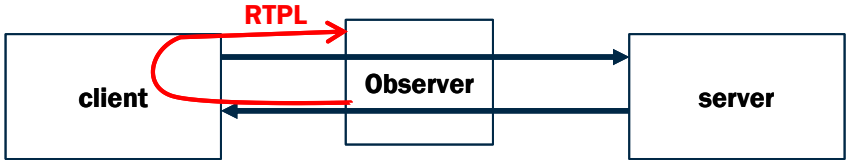
Roundtrip Packet Loss: 2 implementations

- 1 bit Packet Loss Marking (with SpinBit):
  - ¼ flow monitoring («slowest direction» packet rate)
    - Client Generation phase: 2 RTT
    - Pause: 2 RTT
    - Client Reflection Phase: ≈2 RTT
    - Pause: 2 RTT
- 2 bit Packet Loss Marking (It works also if SpinBit is disabled):
  - ½ flow monitoring («slowest direction» packet rate)
    - Client Generation phase: 100 ms.
    - Client Reflection Phase: ≈100 ms

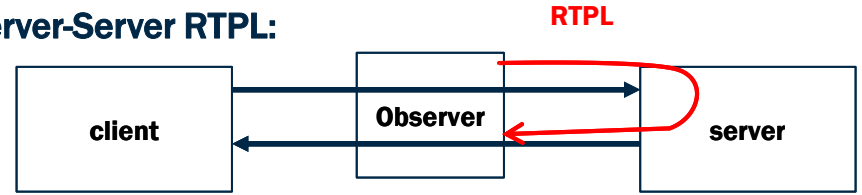
2Point RoundTrip Packet Loss properties (1 observer)



- **Observer-Client RTPL:**

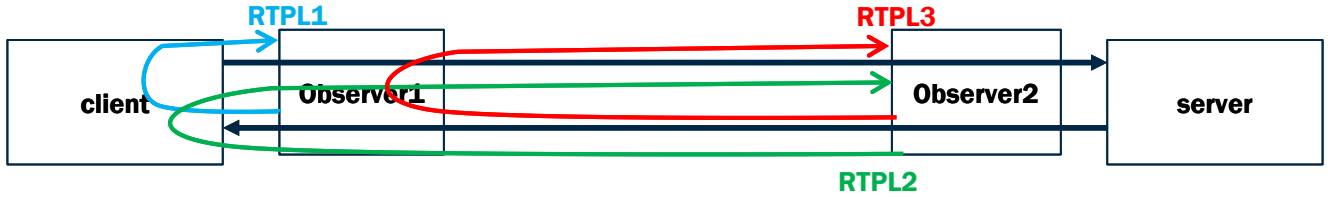


- **Observer-Server RTPL:**



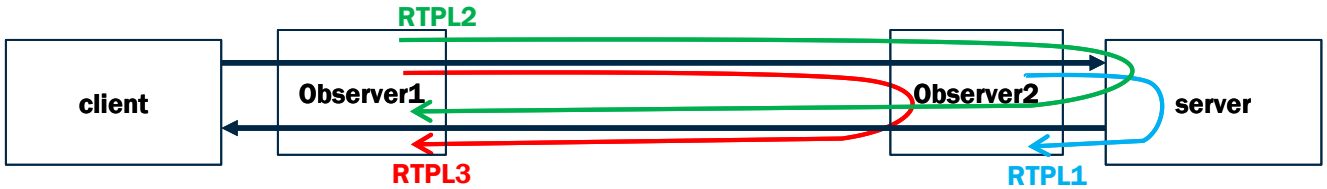
2Point RoundTrip Packet Loss properties (2 observers)

- Observer2-Observer1 RTPL:



Observer2-Observer1 Round-Trip: RTPL2 - RTPL1 = RTPL3

- Observer1-Observer2 RTPL:



Observer1-Observer2 Round-Trip: RTPL2 - RTPL1 = RTPL3

RTPL vs QL measurements (draft-ferrieuxhamchaoui-tsvwg-lossbits)

QL measurements: strengths and weaknesses

+ Q bit measurement analyzes all production traffic (RFC8321).
+ Very simple implementation.
+ Ongoing on field experimentation.
- Q bit signal enables end-to-observer measurement (not end-to-end).
- L bit notifies packet retransmissions not losses (so it measures only 1 direction in most cases) and measurement accuracy is influenced by marked packet losses.

RTPL: strengths and weaknesses

+ It can require just 1 bit instead of 2 (so in QUIC the third bit is available for other uses, e.g. Delay bit).
+ The observer can measure Round Trip Packet Loss by monitoring only one direction.
+ All the explicit round trip measurement properties are enabled (slides: 14-15).
- Analysis of a percentage of production traffic (in our implementation: ¼ or ½ of the packet rate of the slowest direction).
- The implementation changes in case the spin bit is enabled or not (at the current state of development).

Thank you