New Spin bit enabled measurements with one or two more bits

Draft-cfb-ippm-spinbit-new-measurements-00

Prague, March 2019, IETF 104 – IPPM WG

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

Spin Bit: Why?

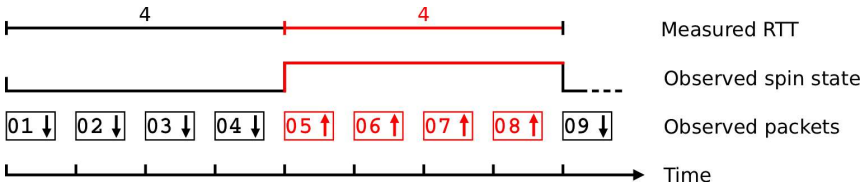
IPPM document: draft-trammell-ippm-spin

- The Spin Bit methodology, in a client–server protocol, allows an explicit per-flow transport-layer signal for hybrid measurement of end-to-end RTT.
- The Valid Edge Counter (VEC) validates the correct Spin Bit measurements in case of impairments (out of sequence, losses, traffic holes...).

How Spin Bit works

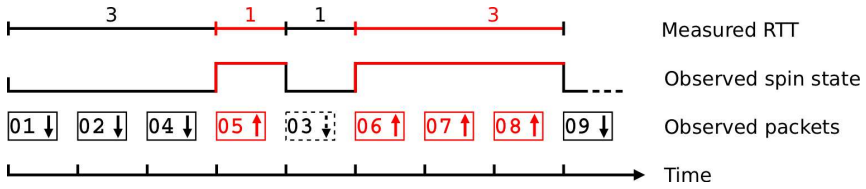
- The spin bit is a simple enhancement of a client-server protocol that causes one bit in the header to ‘spin’, generating one edge (a transition from 0 to 1 or from 1 to 0) once per end-to-end RTT.
- Any device on path can then measure the time (on its local clock) between these edges to generate one 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



Spin Bit limitations

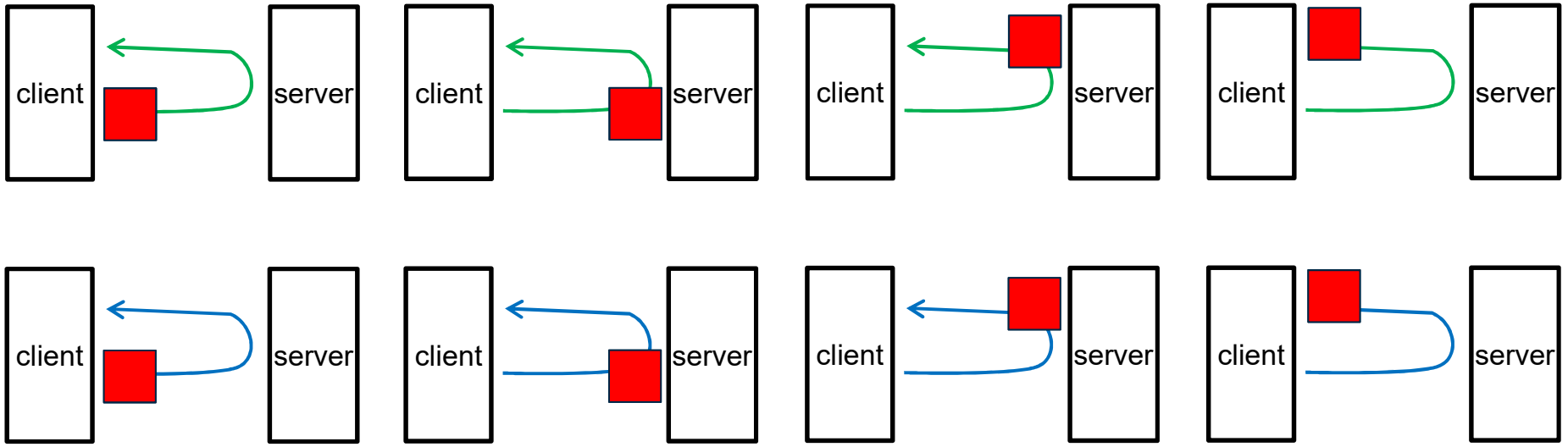
- Packet loss will tend to cause wrong estimates of RTT due to period width changes.
- Reordering of a spin edge will cause drastic underestimates of RTT since it will cause multiple edges to be observed per RTT. So we need an extra instrument to correctly recognize periods, eluding overlapping.



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

Delay Sample

- The idea is to have a single packet, with a second marked bit, called «Delay Bit», that bounces between client and server. This is the Delay Sample (DS).
- Only one Delay Sample «inside» each Spin Bit period (created by the Client when the measurement starts and regenerated by the Client only when the Delay Sample is lost).
- The Delay Sample is a reference for every round trip calculation (in addition to spin bit signal).



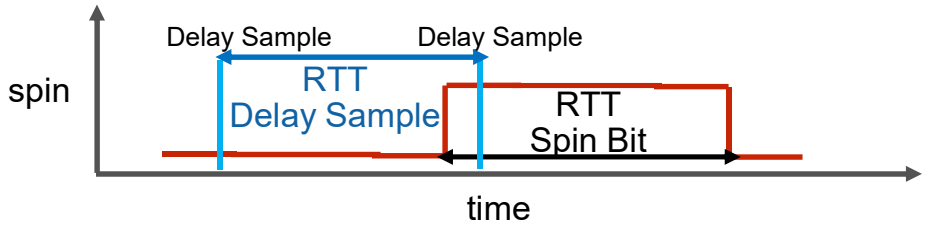
Delay Sample red, Spin Bit green or blue

1) Delay Sample: how the Double Marked sample works

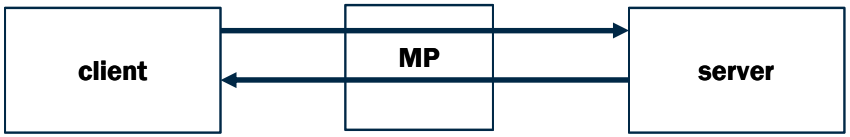
- **When the measurement starts**, and the Spin Bit is set to 1 (or 0) by the client, it also sets the Delay Bit of the first packet to 1, so it becomes the Delay Sample for this marking period. Only this packet is marked with the Delay Bit=1 for this marking period.
- **Delay Sample reflection**: when a packet with Delay Bit=1 arrives, server/client marks the first packet in the opposite direction as the Delay Sample
- **Client side control**: if a marking period ends without a Delay Sample, the client waits a further empty period. It starts again to mark the delay sample the following Spin Bit period. The empty period is needed to signal to the intermediate points that there was an issue and a new delay measurement session is starting.

2) How Delay Sample improves Spin Bit mechanism

- **Key Goal: stabilize RTT measurements influenced by packet loss and reordering**
- 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 should give 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.
- End Points (client and server) instead, use the packet sequence number (encoded inside each packet) to avoid reordering problem and, in case of client perspective, regenerate the Delay Sample when an "empty" period is recognized.

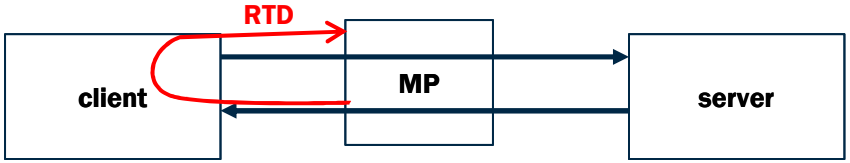


2Point Round-Trip Delay (1)

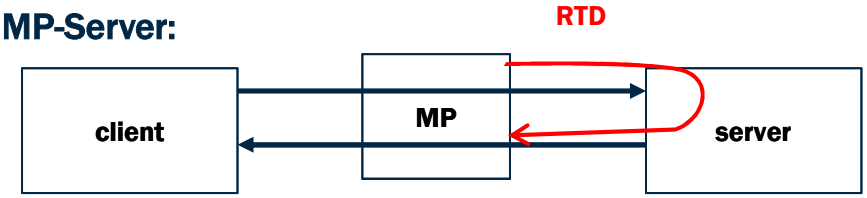


MP= Measurement Point (Observer)

- **RTD MP-Client:**

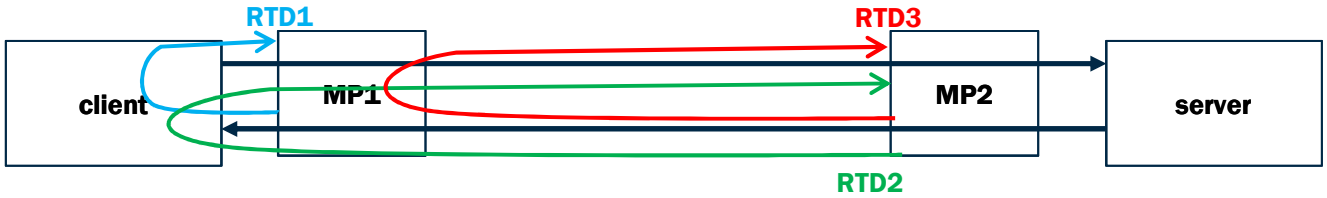


- **RTD MP-Server:**



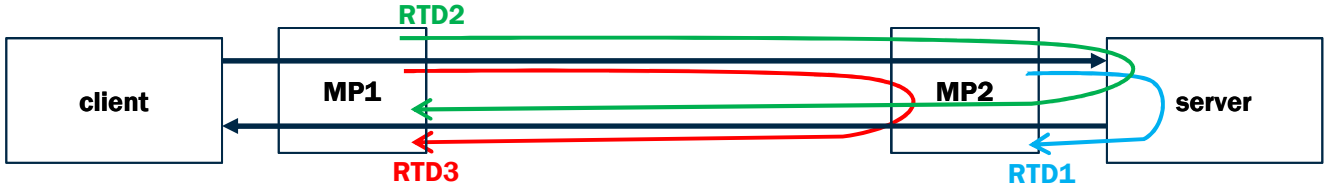
2Point Round-Trip Delay (2)

- RTD MP2-MP1:



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

- RTD MP1-MP2:



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

VEC and Delay Sample compared

VEC: strengths and weaknesses

+ Produces one more valid periods for each edge loss (it's quicker on restart).
+ Observer implementation simpler than Delay Sample (no timer).
- Requires three bits (the entire amount made available by Google for experimentations).
- Decreases its performance in the presence of packet reordering (discarded measurements).

Delay Sample: strengths and weaknesses

+ Requires only two bits, leaving the third one available for other uses (e.g. Packet Loss).
+ Produces more valid periods in case of packet reordering (it does not discard periods and produces corrected measurements).
- Produces less valid periods in case of losses (slower on restart when a delay sample is lost or delayed: an empty period is left).
- Observer implementation needs a timer, the waiting period, to skip false periods in case of packet reordering (the waiting period duration is a tradeoff because it is also the minimum measurable RTT).

RoundTrip Packet Loss

- We define a new performance metric, the **RoundTrip Packet Loss**.
- The Client marks a train of packets, this packets bounces between Client and Server to complete 2 rounds, an Observer counts the marked packets during the 2 rounds and compare numbers to find losses.

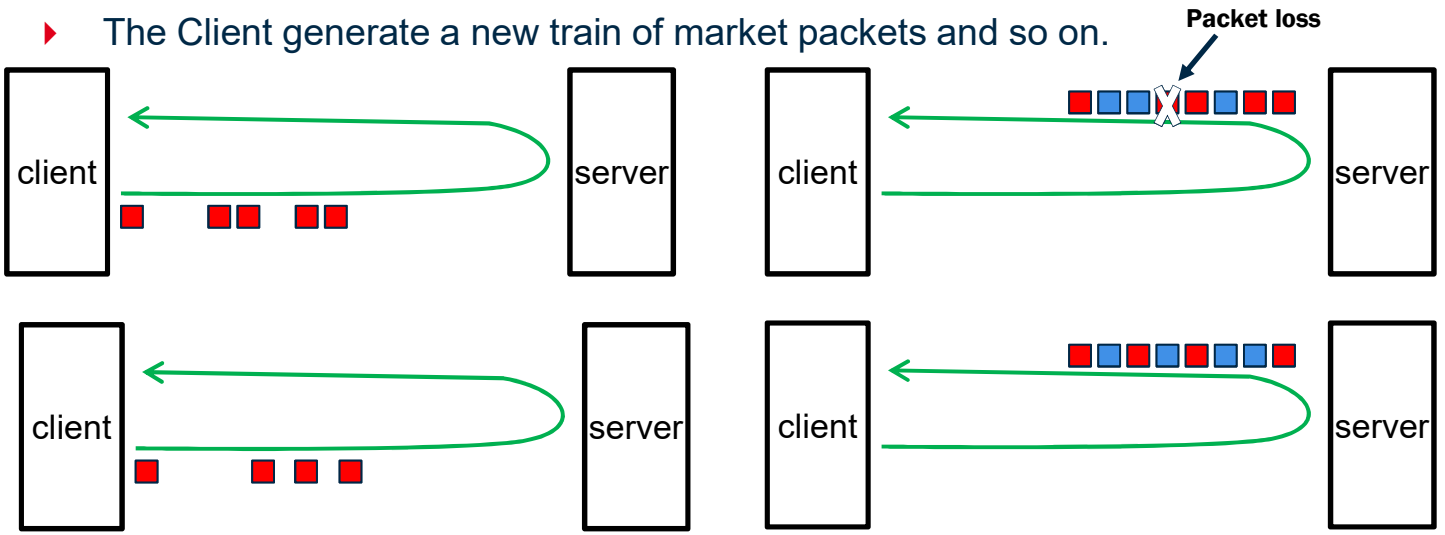
Problem: how many packets to mark to avoid marked packets congestion on the slowest traffic direction?

Solution: the number of packets that transit on the slowest direction during an RTT.

The main idea: Roundtrip Packet Loss (1)

- The Client generate a train of market packets (Packet Loss bit)
- The Server reflects these packets (it inserts some not marked packets).
- The client reflects the marked packets.
- The server reflects the marked packets
- The Client generate a new train of market packets and so on.

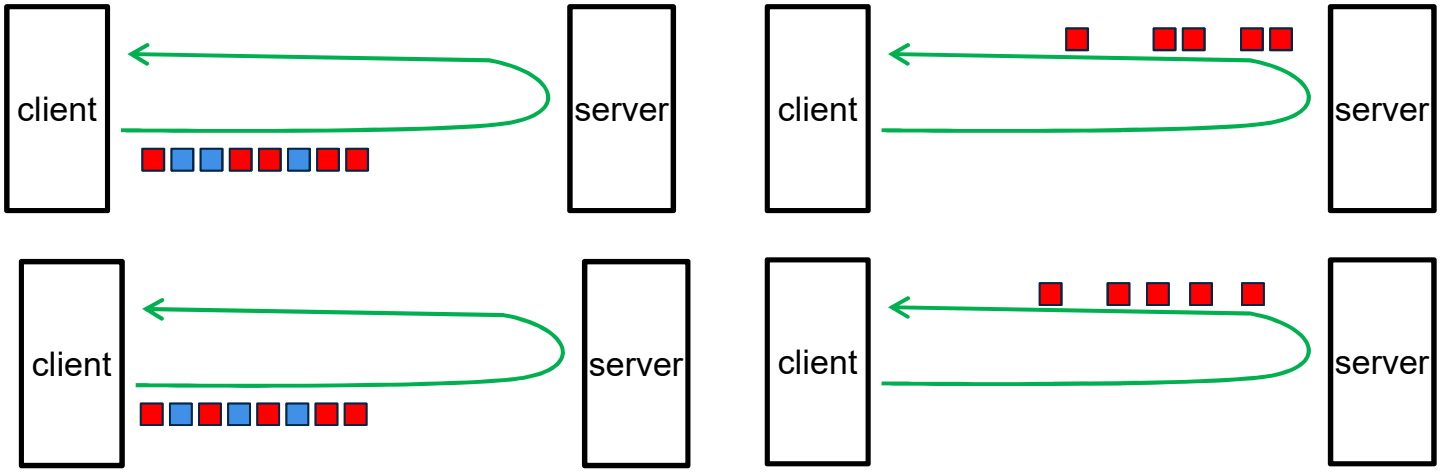
Upload flow has less packets



Marked packets red, Empty packets blue

The main idea: Roundtrip Packet Loss (2)

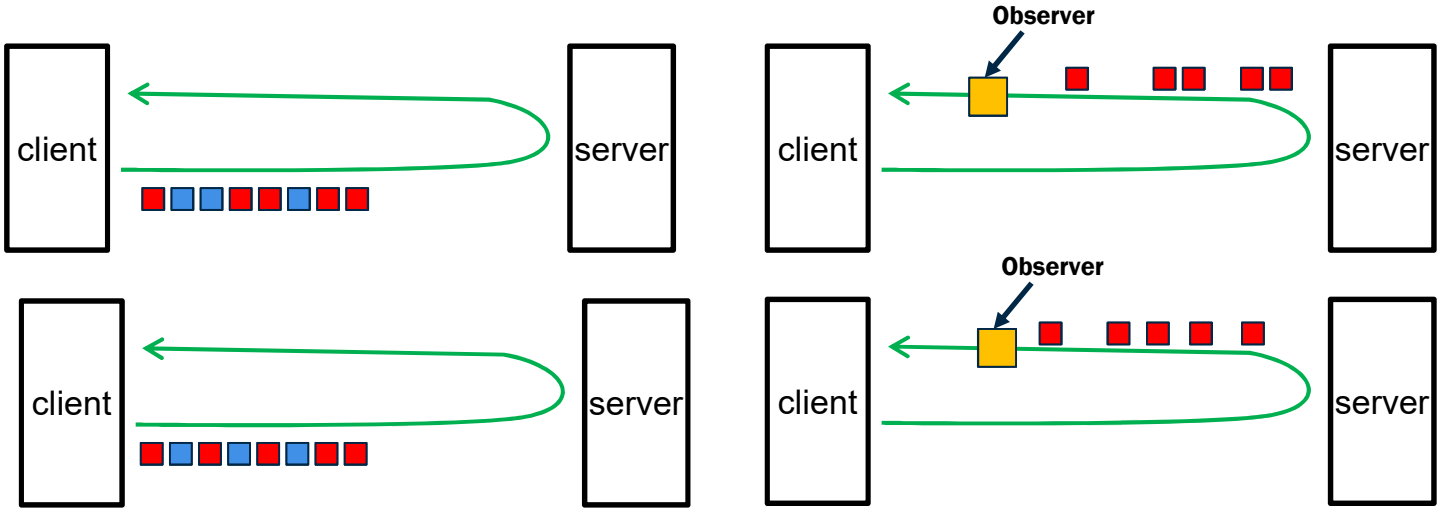
- If upload flow has more packets than download flow (no packet loss):



Marked packets red, Empty packets blue

Roundtrip Packet Loss: 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».



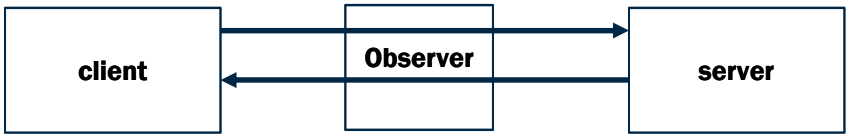
Marked packets red, Empty packets blue

Roundtrip Packet Loss: efficency

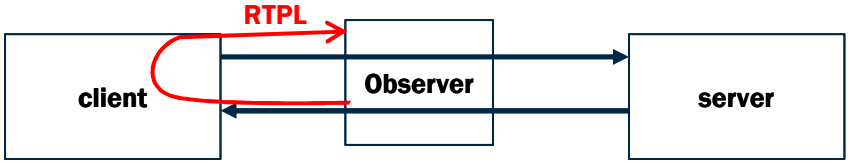
- 1 bit Packet Loss Marking: ¼ flow monitoring («slowest direction»)
  - Client Generation phase: 1 RTT
  - Pause: 1 RTT
  - Client Reflection Phase: 1 RTT
  - Pause: 1 RTT
- 2 bit Packet Loss Marking: ½ flow monitoring
  - Client Generation phase: 1 RTT
  - Client Reflection Phase: 1 RTT

Marked packets red, Empty packets blue

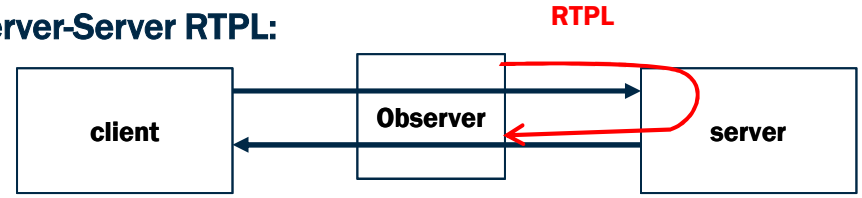
2Point RoundTrip Packet Loss (1)



- **Observer-Client RTPL:**

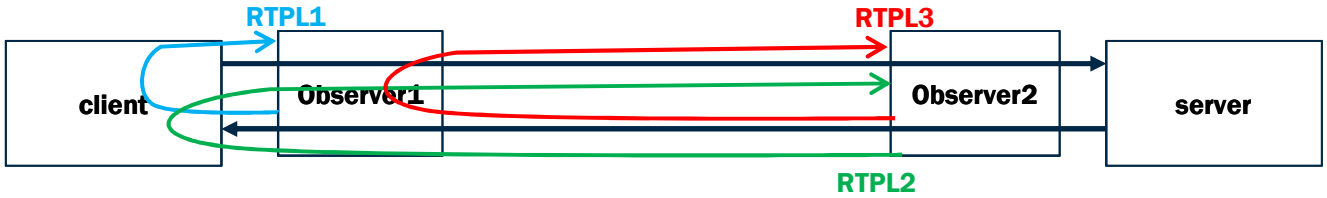


- **Observer-Server RTPL:**



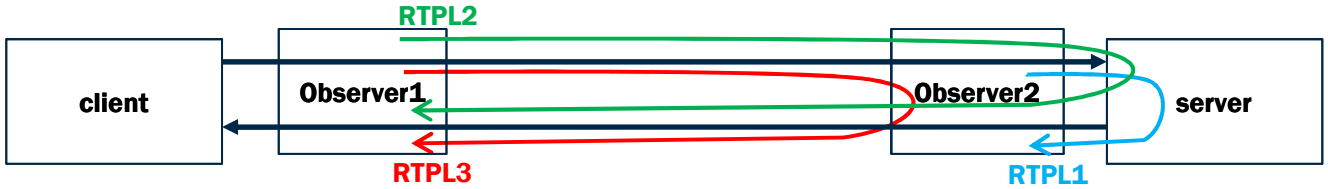
2Point RoundTrip Packet Loss (2)

- Observer2-Observer1 RTPL:



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

- Observer1-Observer2 RTPL:



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

Thank you