Spin bit and beyond @ Orange

A.Ferrieux, I.Hamchaoui

Why do we care ? (1/2)

- Encryption obviously shines in key QUIC goals
- Requesting measurement bits sounds like an uphill battle…
- ... and yet we're here because we 're **scared**

Why do we care ? (2/2)

- ... scared to lose a **cheap and efficient tool** in everyday network troubleshooting: TCP headers

- You may not believe us because you think we have alternatives; **we don't** !

It's a less than ideal world

- Our networks include **legacy hardware**
- **Responsibility boundaries** come into play
- ... but still, why don't we manage with **local inspection** ?

Alternatives that don't work (1/3)

- **Drop counters** on switches and routers ?
  - often hard to reach (could be a leased network)
  - often inaccurate (don't include internal fabric overflows)
  - hard to correlate finely with other events
  - coarse granularity (per interface)

Alternatives that don't work (2/3)

- Two-point **segment bracketing** ?
  - very expensive
  - cannot run continuously without extreme precaution
  - packet correlation may be hard

Alternatives that don't work (3/3)

- Using only **active probes** ?
  - **moving** them around is very expensive
  - **reproducibility** is not guaranteed
  - cannot be inserted in the middle of **tunneled** segments (e.g. GTP in mobile networks) or pure-L2 paths

So, what does work ?

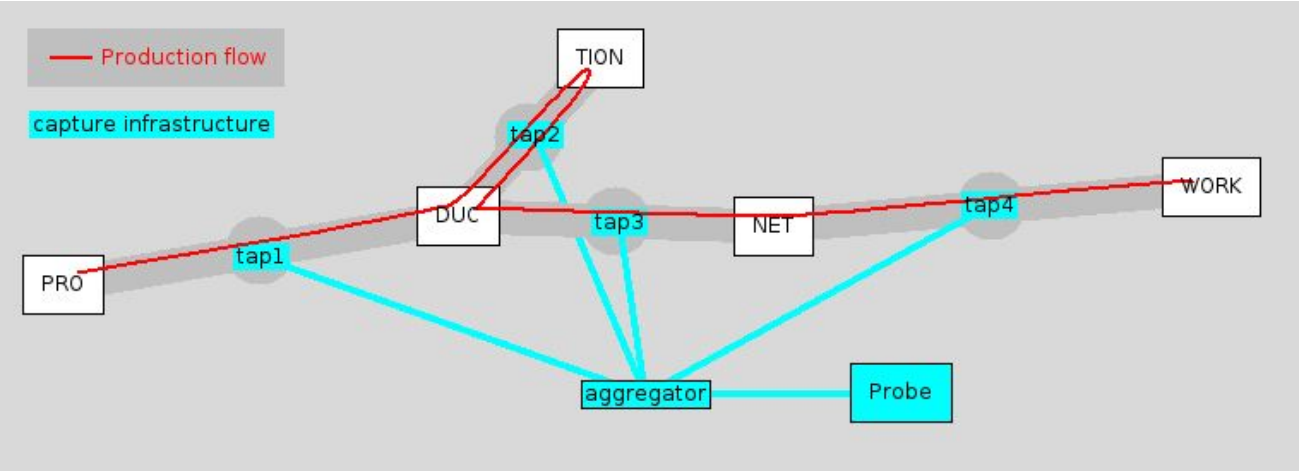
- **Dichotomy on loss and latency** is the only efficient tool !
  - with TCP headers, we get RTT and loss contributions on **either side** of a capture point
  - we then **quickly home in** on the offending box
  - this still works with **multiple offenders** (which happens)

Not a toy ! (1/3)

- We need to do this dichotomy **very frequently**:
  - ever-evolving networks
  - new affiliates
  - problems outside our responsibility => better locate them!

Not a toy ! (2/3)

- ... so we **invest hardware and manpower** into it:
  - passive probes doing traffic capture and real-time analysis
  - extensive deployment of capture points (optical taps and aggregators)



Not a toy ! (3/3)

- ... and the **ROI is good**: we solve real problems:)
  - misbehavior of core components under load
  - access network bottlenecks
  - "not guilty": loss or RTT proven to be **in another AS**

Doing the same with QUIC ?

- We have a proposal **fitting into the 3 Reserved bits** in draft-14 (1st octet of short headers):

  0K110SQE

  S = Spin bit => gives **half-RTT on either side** of the capture point

  Q = sQuare sequence bit => gives **upstream loss**

  E = E2E bit => gives end-to-end **loss**, and **downstream** by difference

  => the full dichotomy lives on!

How does it work ? (1/3)

- **S** = Spin bit => gives **half-RTT on either side** of the capture point
  - see **draft-trammel-quic-spin03**
  - just the spin bit, **not the VEC** (we need the other 2 bits)
  - see Marcus Ihlar's **heuristics** to do without VEC

How does it work ? (2/3)

- **Q** = sQuare sequence bit => gives **upstream loss**
  - proposed by Kazuho during early discussions on loss measurement
  - principle:
    - a square signal of a **well-known fixed number** of packets
    - the observer counts packets
    - any difference with the well-known period indicates upstream loss

How does it work ? (3/3)

- **E** = E2E loss bit => gives end-to-end **loss**, and **downstream** by difference

    - refinement of the loss bit idea:
        - receiver keeps track of recent losses
        - one outgoing packet marked with E=1 ⇔ one loss identified earlier
        - keep marking E=1 until all recent losses reported
        - loss rate increases => more ACKs => more packets to carry E=1

=> in most cases, the total number of E=1 equals the **E2E loss count**,
and they are reported **rather timely**

=> the complexity cost for the endpoints is **very low**

Demo

- Implemented in PicoQuic:

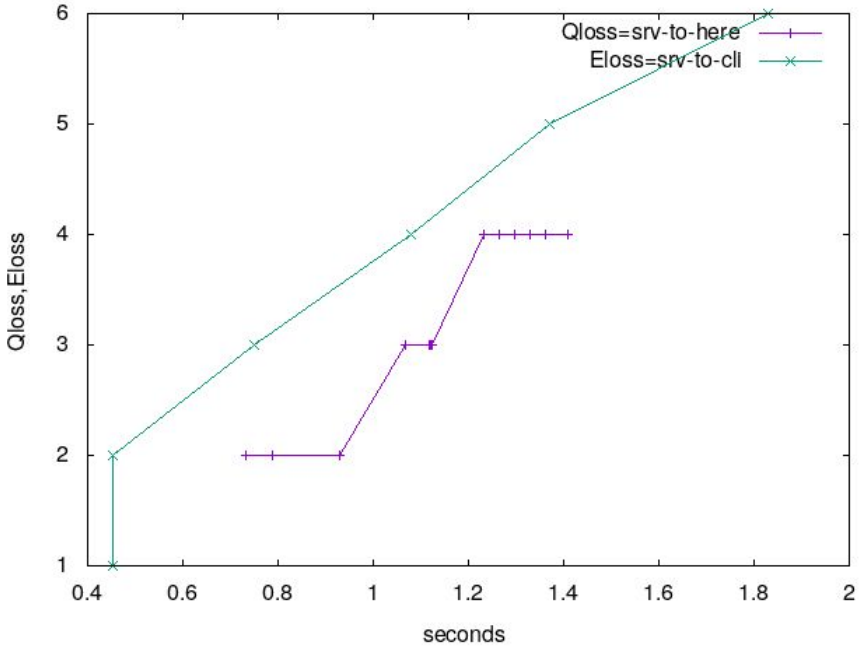
  https://github.com/private-octopus/picoquic/compare/master...ferrieux:master

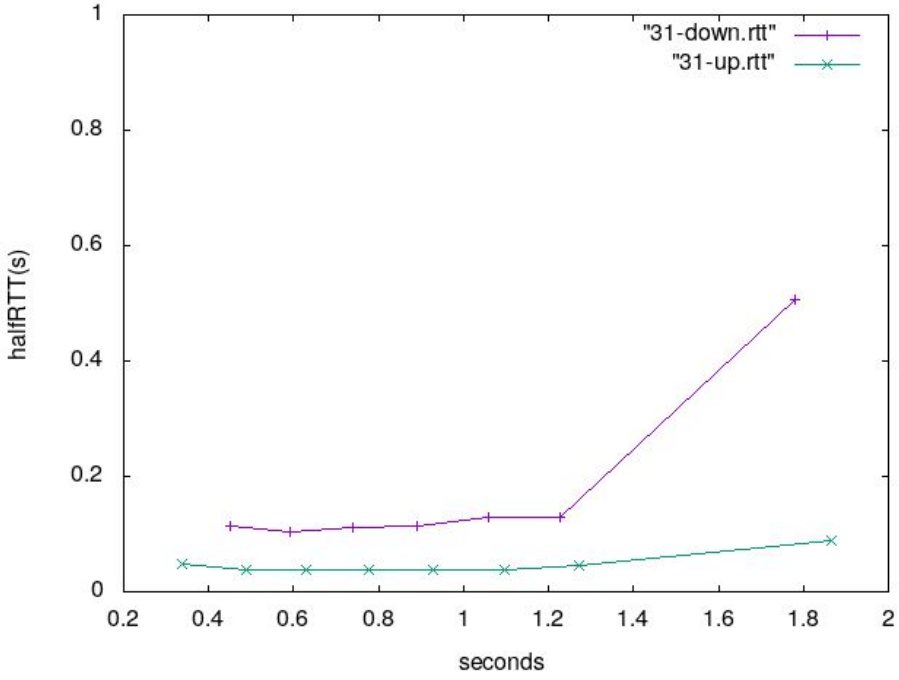
- Online analysis tool available, just upload your pcap:

  https://193.252.113.227/cgi-bin/quicspin.cgi

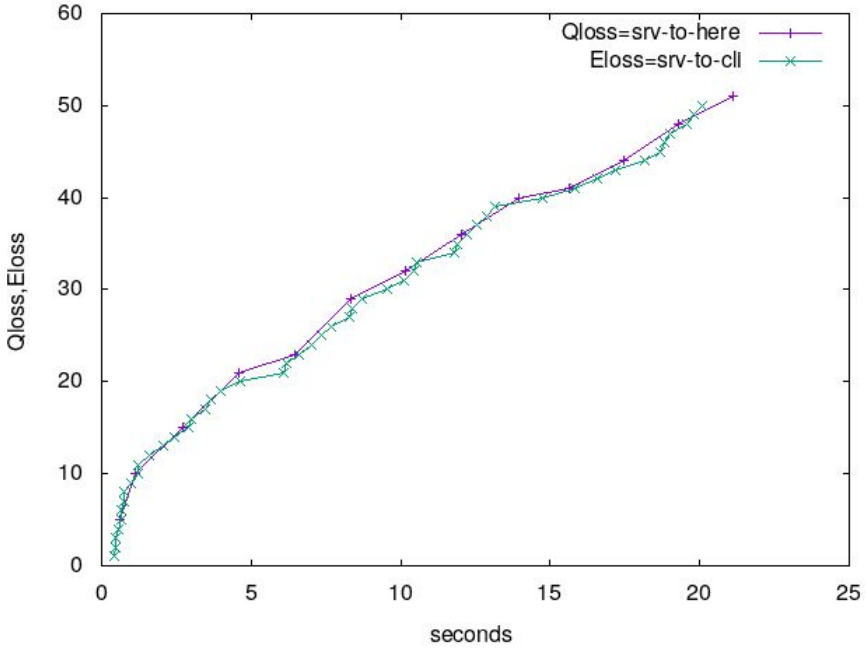
- Unit tested in many scenarii on real networks.

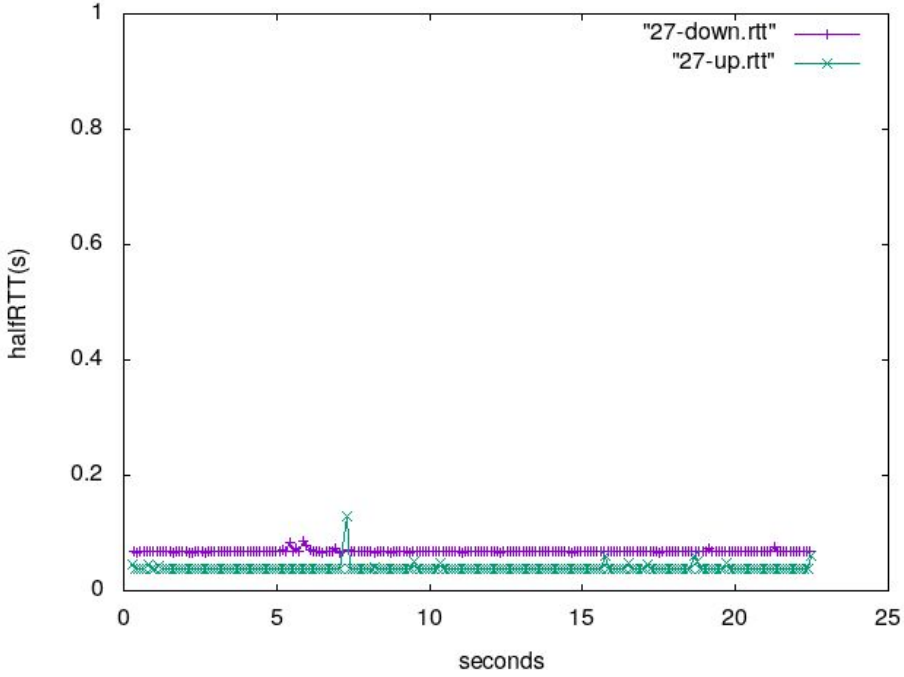
Demo: little loss, RTT buildup



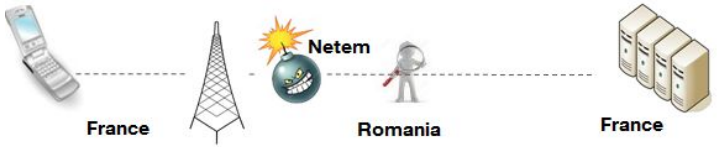


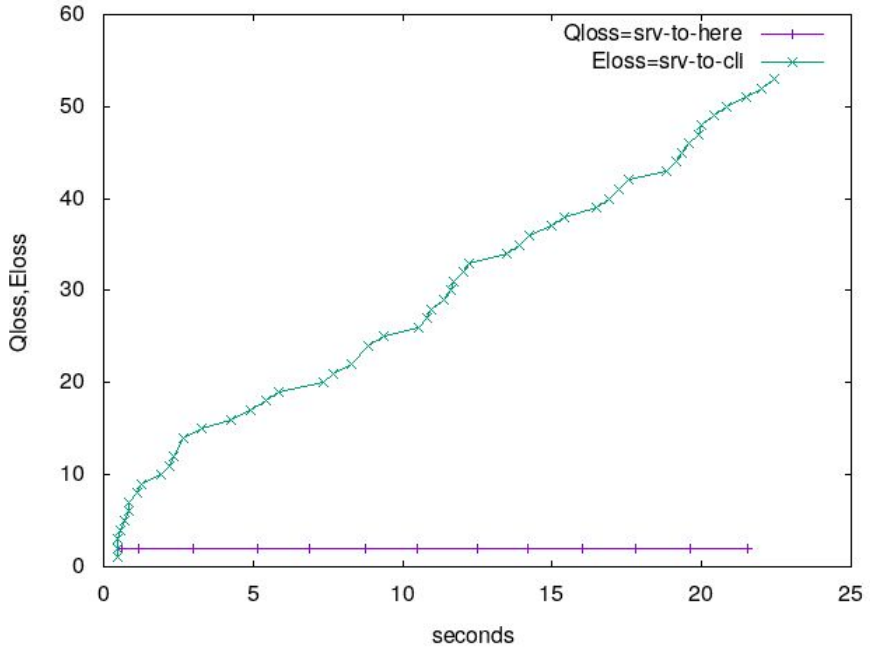
Demo: loss above, no RTT buildup

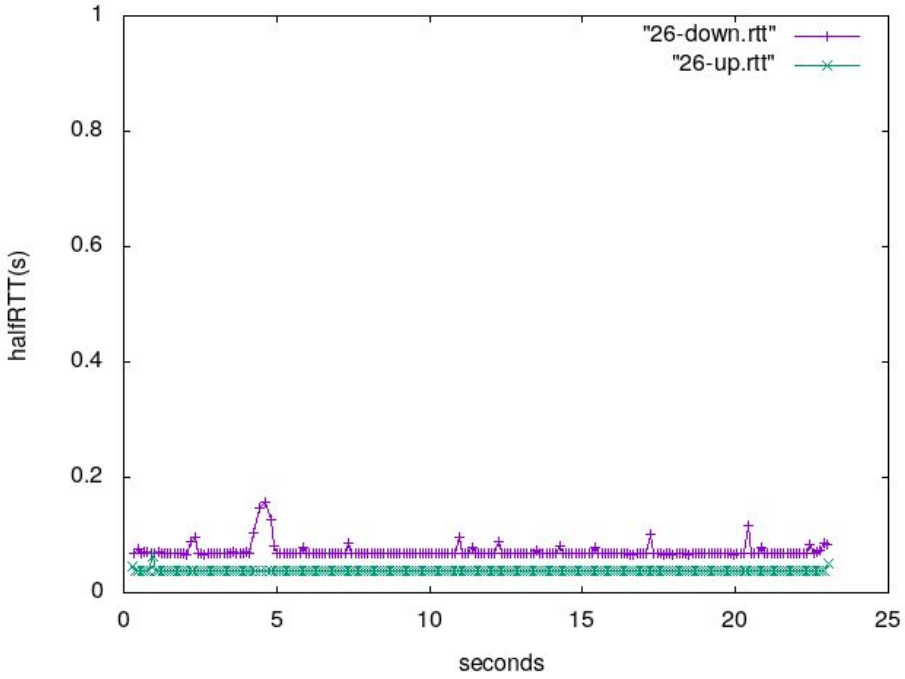




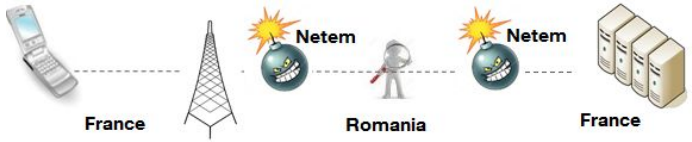
Demo: loss below, no RTT buildup

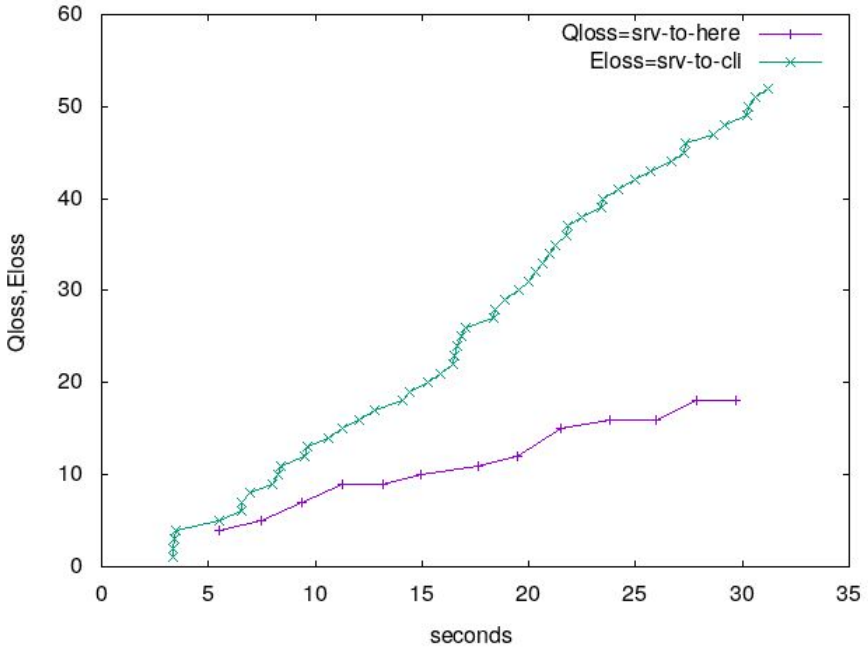


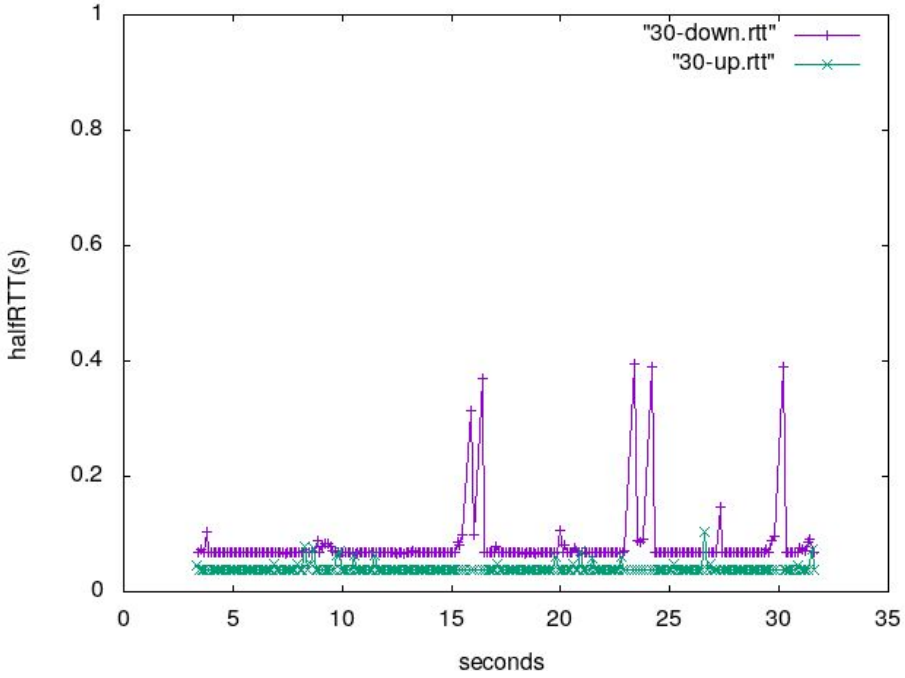




Demo: loss on both sides, no RTT buildup



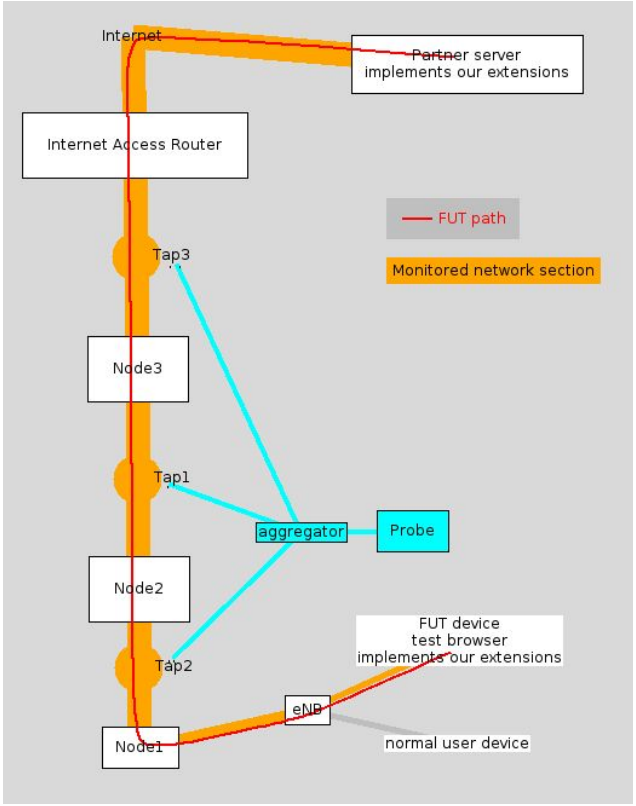




What next ? (1/2)

- Plan A: a **FUT with partners** providing Browser and Server
  - assumes these 3 bits remain available as per the QUIC spec
  - will be run on a production network with multiple capture points
  - will allow the exact same dichotomy as today with TCP, **on the full path**
  - OK with **non-rooted devices**
  - requires to **find partners**

Plan A = full path ; modified browsers + servers

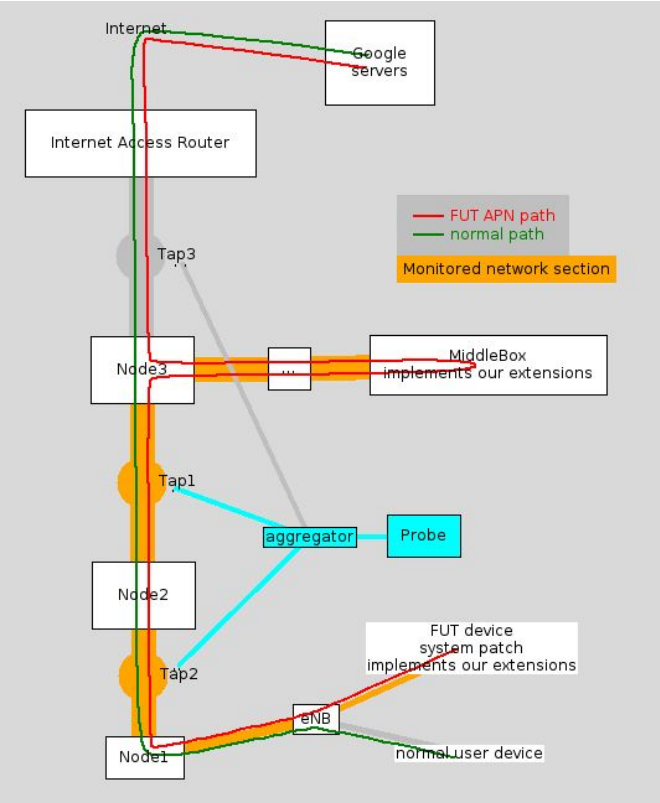


What next ? (2/2)

- Plan B: a **FUT with client system patch** and middlebox
  - assumes nothing, works with non-IETF gQUIC (**L3 header "trick"**)
  - LD_PRELOAD or iptables module on the client (Android or Linux)
  - iptables module on the middlebox
  - will be run on the same production network
  - will allow the exact same dichotomy as today with TCP, **restricted to the segment** between client and middlebox.

  - **vanilla Chrome and Youtube** clients talking to vanilla Google servers
  - **rooted devices** + specific config to go through middlebox
  - **small network segment**

Plan B = small segment ; vanilla Chrome/Youtube



Annex: End-to-end loss