On Explicit In-Band Measurement

Brian Trammell and Mirja Kühlewind, ETH Zürich
Based in part on ongoing work with Mark Allman (ICSI) and Rob Beverly (NPS)
In the beginning…

- ...there was ping, and it was good.
- (still the only explicit measurement facility in the stack.)
- Periodic measurement via cron
- Visualization and storage with rrdtool
- Distributed measurement via telnet
- Distributed measurement via ssh
- Glue everything together with perl

Actually, this is pretty much SmokePing.
Today’s talk

- A couple of questions:
  - what if we’d designed measurement as a first-order service of the protocol stack?
  - how far from that ideal are we, really?
- A proposed answer:
  - design a measurement layer into the stack
  - use deployable technologies to make it work
- Some shameless promotion:
  - H2020 MAMI project, working to make this a reality
Everything after ping is a hack

- And even ping doesn’t work that well:
  - ICMP blocked, different codepaths, ECMP routing.
- Traceroute: overload ICMP Time Exceeded messages to infer Layer 3 topology
  - Same problems as ping, but ECMP is worse.
- TCP throughput testing: how many bytes sent / sec?
  - Unreliable as an indication of network conditions [1].
- Netflow/IPFIX: watch the flows go by and measure
  - Passive RTT measurement [2] broken by ACK optimizations [3], etc.
  - Inflexible, low-rate sampling, even though we know better [9].
Let’s ask a different question…

- **What if we had designed measurement into the stack as a first-order service?**
  - “Big five” metrics (loss, latency, jitter, rate, reordering) as socket properties, with transport MPI/API for access.
  - You don’t need much more for QoE-relevant network metrics
  - Header fields explicitly defined for measurability
    - Constant-rate timestamps for latency/jitter
    - Transport-independent exposure of loss/reordering
    - Exposure in header allows passive as well as endpoint measurement
    - Detection of header manipulation (required for dynamic transport selection)
    - Explicit endpoint control over measurement exposure
  - But we didn’t, so **how can we get there from here?**
How close are we to the goal?

- TCP seq/ack number analysis for loss/reorder?
  - Always exposed, and roundly abused in the Internet
  - Only works with TCP
- TCP TSOPT timestamps for latency/jitter
  - Only works with TCP, enabled on about 30% of hosts
  - No application hooks for \textit{explicit} enablement
  - Need heuristics to estimate sender clock rate
- Checksums provide poor header manipulation detection
How close are we to the goal?

Modern networking stacks are heavily instrumented

- netstat -s -p tcp on OSX yields 82 event counters.

Application instrumentation also includes collection
- e.g. telemetry.mozilla.org

Phase 1: generalizing and standardizing access to data we already have.
- e.g. mPlane [4]
A Measurement Layer

- Phase 2: define a “measurement layer” for explicit exposure of information as part of normal protocol exchanges.
  - e.g. IPv6 PDM DO [5], HICCUPS [6].
  - You don’t have to instrument every packet, every endpoint, or every router to get much better information than we have today.
A Measurement Layer

- Insight: shifting the burden to analysis-time reduces the runtime burden.
- Cumulative nonce ($n_{tx}, \sum n_{rx}$) added to each / sample of packets [8] allows loss rate estimation.
- Timestamp echo ($t_{tx}, t_{rx}, t_{\Delta rx}$) with constant-rate clock [7] and remote delta allows latency and jitter estimation.
- Protected header hash echo ($h_{tx}, h_{rx}$) allows detection of header manipulation [6].
  - Shared-secret protected hashes allow secure detection by endpoints
  - Unprotected hashes detect only accidents
- Insight: Each of these can work at low sampling rates for large flows.
  - How much smarter can we be for less than one bit per packet?
Sounds great. Let’s do it!

- Now we just have to find the bits…
- IPv6 Destination Options?
  - not very deployable, may be nearing deprecation, v6 only.
- IPv4 options?
  - even less deployable, v4 only.
- in the TCP header?
  - Options hard to deploy, TCP only, measurement not properly layer 4.
  - HICCUPS reclaimed a few bits from the header itself

- **Adding new layers to the stack is hard.**
Adding new layers to the stack for fun and profit

- Our “measurement layer” is a special case of a more general problem:
  - Internet layer is hop-by-hop, stateless
  - Transport layer is end-to-end, stateful
  - Where do all of the complex, stateful, not necessarily end-to-end functions we’ve built go?
- Since we already have a “path layer”, let’s make it explicit:
  - Encryption of transport layer and above to enforce end-to-end-ness
  - Explicit exposure from endpoints to the path of appropriate information
How to implement a path layer

• You can’t add a new layer that today’s routers won’t route.
• NAT: hard* to deploy protocols other than TCP or UDP
• Need to support userland implementation for experimentation and early deployment.
• Conclusion: “path layer” headers as shim over UDP
• Initial findings: 3-6% of Internet hosts may have broken or no UDP connectivity, so we’ll need a backup.
• Define path layer headers so that other future encapsulations (e.g. IPv6 DO) are possible?
Path layer requirements

- Exposure to the path of information, the endpoints decide the path needs
- Cryptographic protection of the rest of the transport layer
- Packets grouping for property binding, on-path state management
- Efficient per-packet signaling
- Integrity protection for exposed headers, allowing modification with endpoint permission
- Protection against trivial abuse of UDP
- Work in progress: draft-trammell-spud-req
Answering the path layer deployability question: a Path Transparency Observatory

- UDP encapsulation sounds great, but decisions about deployment need data.
- Solution: observatory (public release end 2016) to derive common observations about conditions on a given path at a given time.
- Combining disparate measurements leads to better insight.
  - e.g. own measurement data, traceroutes, BGP, traces.
The MAMI Project
Measurement and Architecture for a Middleboxed Internet

- Strong interaction with relevant standards organizations for impact on deployment
- FIRE testbed (MONROE) support for measurement as well as experimentation, especially on mobile broadband access networks
- Learn more at http://mami-project.eu/
In conclusion…

- Two steps to put measurement where it belongs in the stack:
  1. better access to instrumentation we already have
  2. build a layer to expose information explicitly intended for measurement to amplify our ability to measure.
- Adding a layer to the stack for middlebox cooperation gives us the ability to deploy step 2.
  - We believe this is possible atop UDP encapsulation
  - Work in progress: watch IETF SPUD, MAPRG; mami-project.eu.
- Questions? <{trammell, mirja.kuehlewind}@tik.ee.eth.ch>
References

• [1] draft-ietf-ippm-model-based-metrics (IETF IPPM WG Internet-Draft)
• [4] draft-trammell-mplane-protocol (IETF individual Internet-Draft)
• [5] draft-ietf-ippm-6man-pdm (IETF IPPM WG Internet-Draft)
• [7] draft-trammell-tcpm-timestamp-interval (IETF individual Internet-Draft)
Measurement Primitives Illustrated

Cumulative Nonce

Sender

\[ \sum_{n_0}^{n_2} \]
\[ \sum_{n_0}^{n_3, 5} \]
\[ \sum_{n_0}^{n_7} \]
\[ \sum_{n_0}^{n_7, 9} \]
\[ \sum_{n_0}^{n_{12}} \]

Receiver

\[ \Sigma n_0..2 \]
\[ \Sigma n_0..3, 5 \]
\[ \Sigma n_0..7 \]
\[ \Sigma n_0..7, 9 \]
\[ \Sigma n_0..12 \]

Delta Timestamp Echo

Sender

\[ s_{t0}, \text{nil}, \text{nil} \]
\[ \partial_0 \]

Receiver

\[ r_{t1}, s_{t0}, \partial_0 \]
\[ \partial_1 \]

\[ s_{t2}, r_{t1}, \partial_1 \]