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)

mami

measurement and architecture for a middleboxed internet

measurement   **architecture**   **experimentation**

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`

```
Echo or Echo Reply Message

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Data ...
+-+-+-+-+-
```
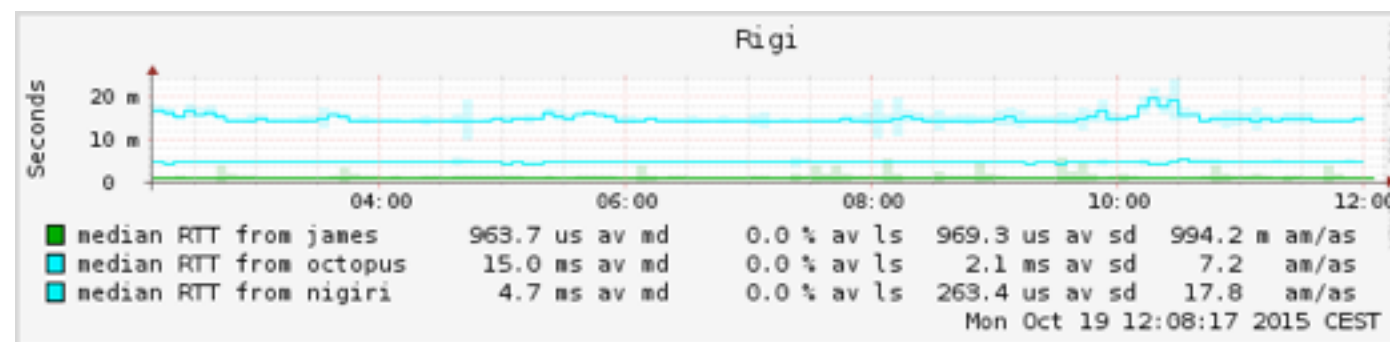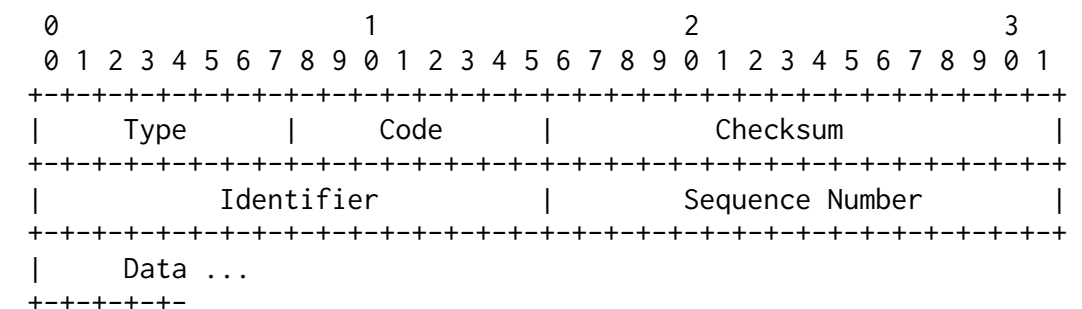

- Actually, this is pretty much SmokePing.

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 *explicit* enablement

  - Need heuristics to estimate sender clock rate

- Checksums provide poor header manipulation detection

How close are we to the goal?

```
% netstat -s -p tcp
tcp:
        136072 packets sent
                36226 data packets (12605543 bytes)
                52 data packets (19892 bytes) retransmitted
                1 resend initiated by MTU discovery
                86569 ack-only packets (49 delayed)
                0 URG only packets
                0 window probe packets
                7894 window update packets
                5277 control packets
                0 data packets sent after flow control
                6 checksummed in software
                        6 segments (339 bytes) over IPv4
                        0 segments (0 bytes) over IPv6
        164742 packets received
                34764 acks (for 12593499 bytes)
                1246 duplicate acks
                0 acks for unsent data
                143462 packets (152392523 bytes) received in-sequence
                62 completely duplicate packets (49185 bytes)
                0 old duplicate packets
                0 received packets dropped due to low memory
                0 packets with some dup. data (0 bytes duped)
                434 out-of-order packets (532085 bytes)
                0 packets (0 bytes) of data after window
                0 window probes
                19 window update packets
                286 packets received after close
                0 bad resets
                0 discarded for bad checksums
                6 checksummed in software
                        6 segments (496 bytes) over IPv4
                        0 segments (0 bytes) over IPv6
                0 discarded for bad header offset fields
                0 discarded because packet too short
        2736 connection requests
        9 connection accepts
        0 bad connection attempts
        0 listen queue overflows
        2611 connections established (including accepts)
        2823 connections closed (including 50 drops)
                96 connections updated cached RTT on close
                96 connections updated cached RTT variance on close
                5 connections updated cached ssthresh on close
        0 embryonic connections dropped
        70310 segments updated rtt (of 31390 attempts)
        83 retransmit timeouts
                0 connections dropped by rexmit timeout
                0 connections dropped after retransmitting FIN
        0 persist timeouts
                0 connections dropped by persist timeout
        40 keepalive timeouts
                40 keepalive probes sent
                0 connections dropped by keepalive
        78 correct ACK header predictions
        126450 correct data packet header predictions
        28 SACK recovery episodes
        2 segment rexmits in SACK recovery episodes
        1454 byte rexmits in SACK recovery episodes
        69 SACK options (SACK blocks) received
        303 SACK options (SACK blocks) sent
        0 SACK scoreboard overflow
        0 LRO coalesced packets
                0 times LRO flow table was full
                0 collisions in LRO flow table
                0 times LRO coalesced 2 packets
                0 times LRO coalesced 3 or 4 packets
                0 times LRO coalesced 5 or more packets
        0 limited transmits done
        28 early retransmits done
        1 time cumulative ack advanced along with SACK
        0 probe timeouts
                0 times retransmit timeout triggered after probe
                0 times fast recovery after tail loss
                0 times recovered last packet
        1606 connections negotiated ECN
                0 times congestion notification was sent using ECE
                21 times CWR was sent in response to ECE
        0 times packet reordering was detected on a connection
                0 times transmitted packets were reordered
                0 times fast recovery was delayed to handle reordering
                0 times retransmission was avoided by delaying recovery
                0 retransmissions not needed
```


HTTP page: Open -> first byte of reply received (ms)
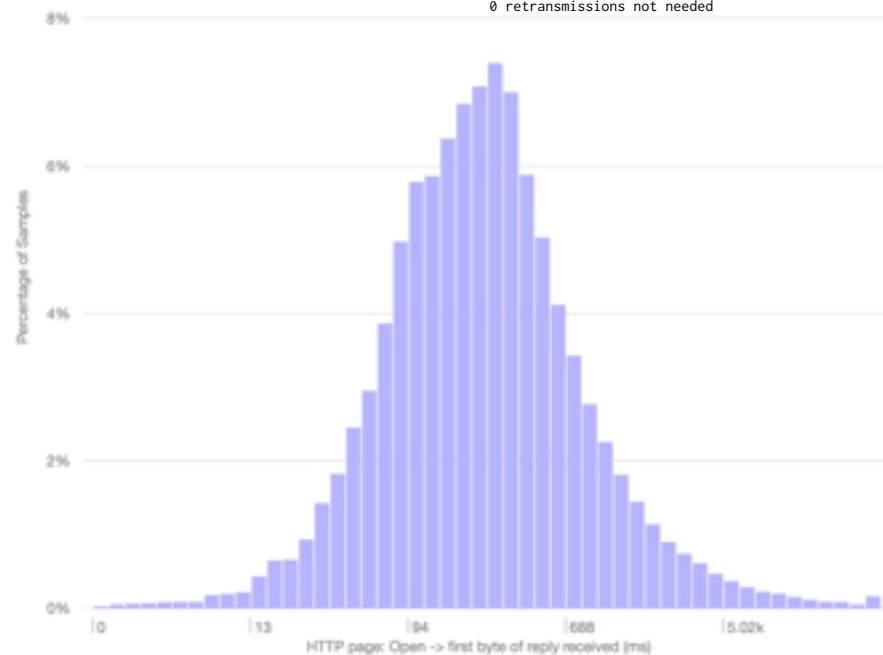
- Modern networking stacks are heavily instrumented
  - `netstat -s -p tcp` on OSX yields 82 event counters.
- Application instrumentation also includes collection
  - e.g. telemetry.mozilla.org
- Phase 1: generalizing and standardizing access to data we already have.
  - e.g. mPlane [4]

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.
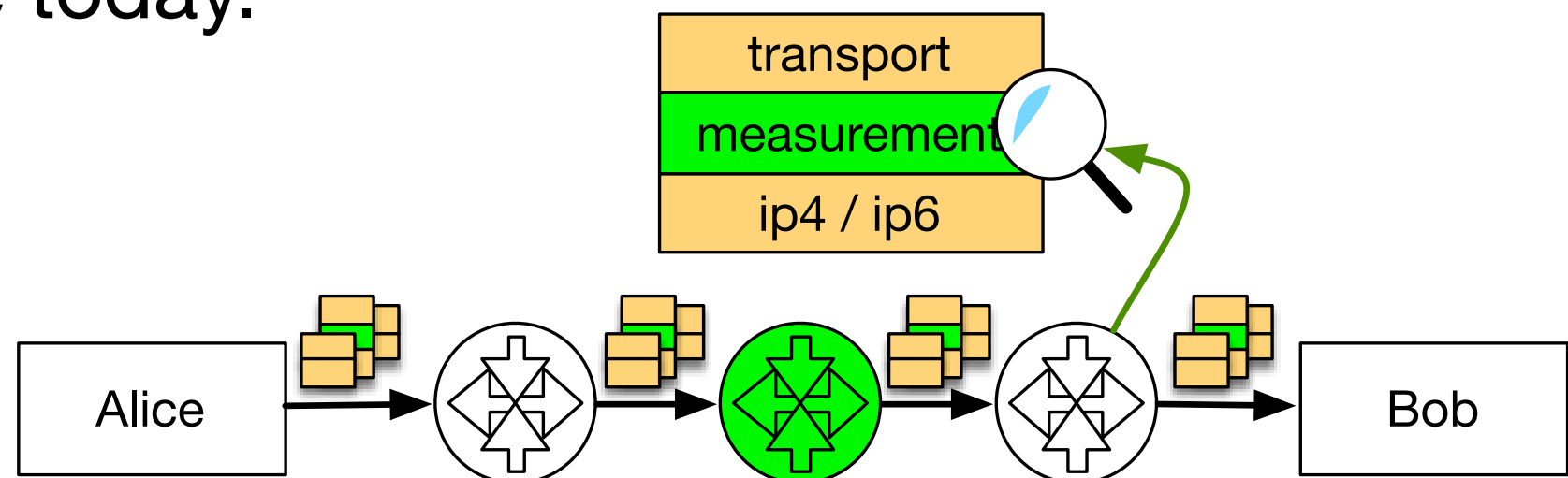
A Measurement Layer

- 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.**

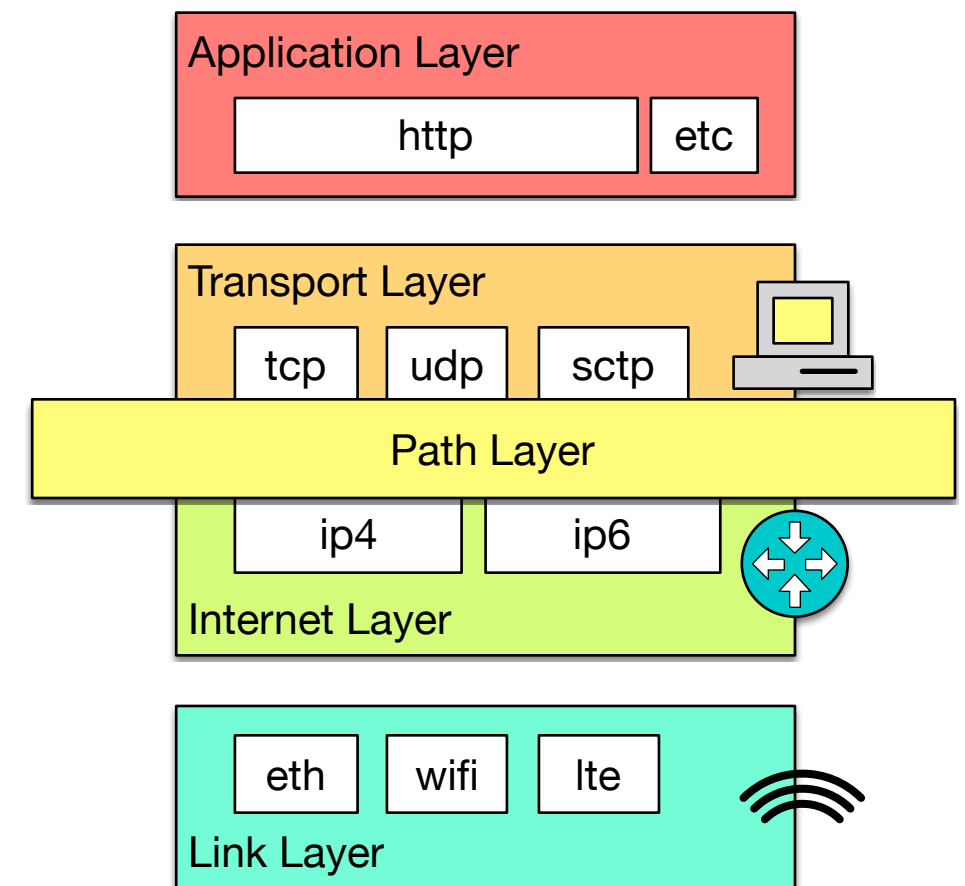 - How much smarter can we be for less than one bit per packet?

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

Application Layer	
http	etc

Transport Layer		
tcp	udp	sctp

Path Layer

Internet Layer	
ip4	ip6

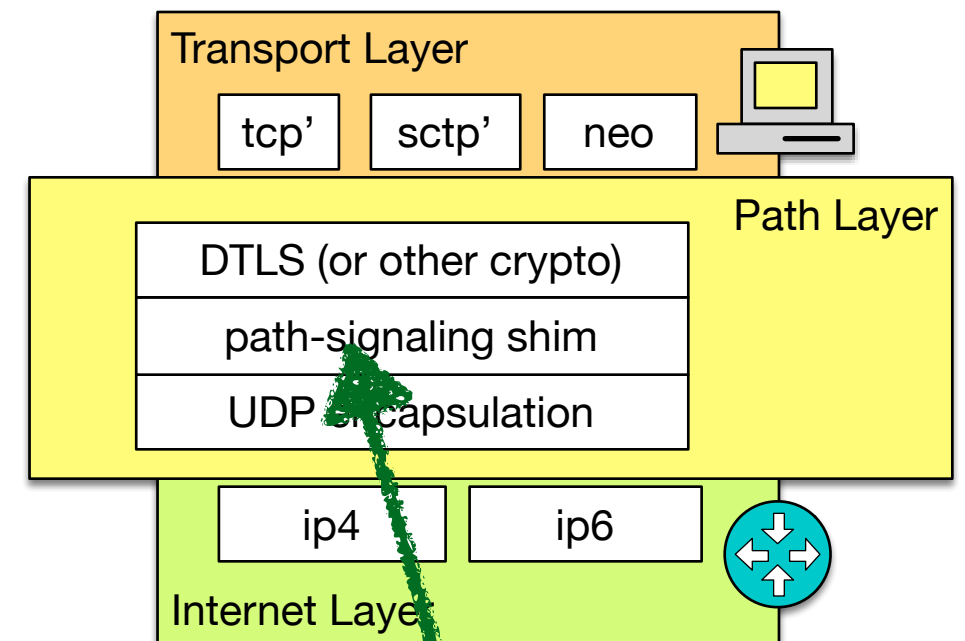| Link Layer		
eth	wifi	lte

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

Transport Layer

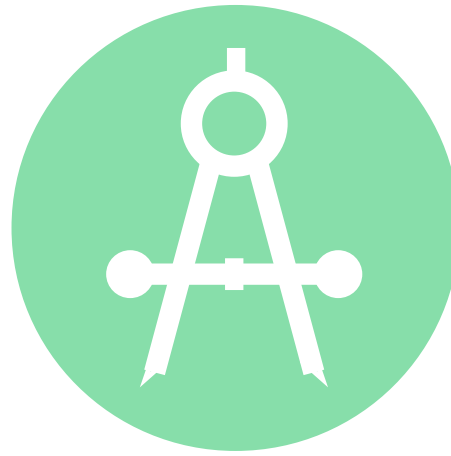tcp' | sctp' | neo

Path Layer

DTLS (or other crypto)

path-signaling shim

UDP encapsulation

ip4 | ip6

Internet Layer

measurement goes here

The MAMI Project

Measurement and Architecture for a Middleboxed Internet

measurement

of deployed middleboxes

architecture

for middlebox cooperation

experimentation

of use case applicability
and deployability

- 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)
- [2] Trammell et al "On Inline Data Integrity Signals for Passive Measurement", TMA 2014
- [3] Ding et al "TCP Stretch Acknowledgements and Timestamps: Findings and Implications for Passive RTT Measurement", Comput. Commun. Rev. 45(3), Jul. 2015.
- [4] draft-trammell-mplane-protocol (IETF individual Internet-Draft)
- [5] draft-ietf-ippm-6man-pdm (IETF IPPM WG Internet-Draft)
- [6] Craven et al "A middlebox-cooperative TCP for a non end-to-end Internet", SIGCOMM 2014.
- [7] draft-trammell-tcpm-timestamp-interval (IETF individual Internet-Draft)
- [8] Savage et al "TCP congestion control with a misbehaving router", Comput. Comm. Rev. 29(5), Oct. 1999.
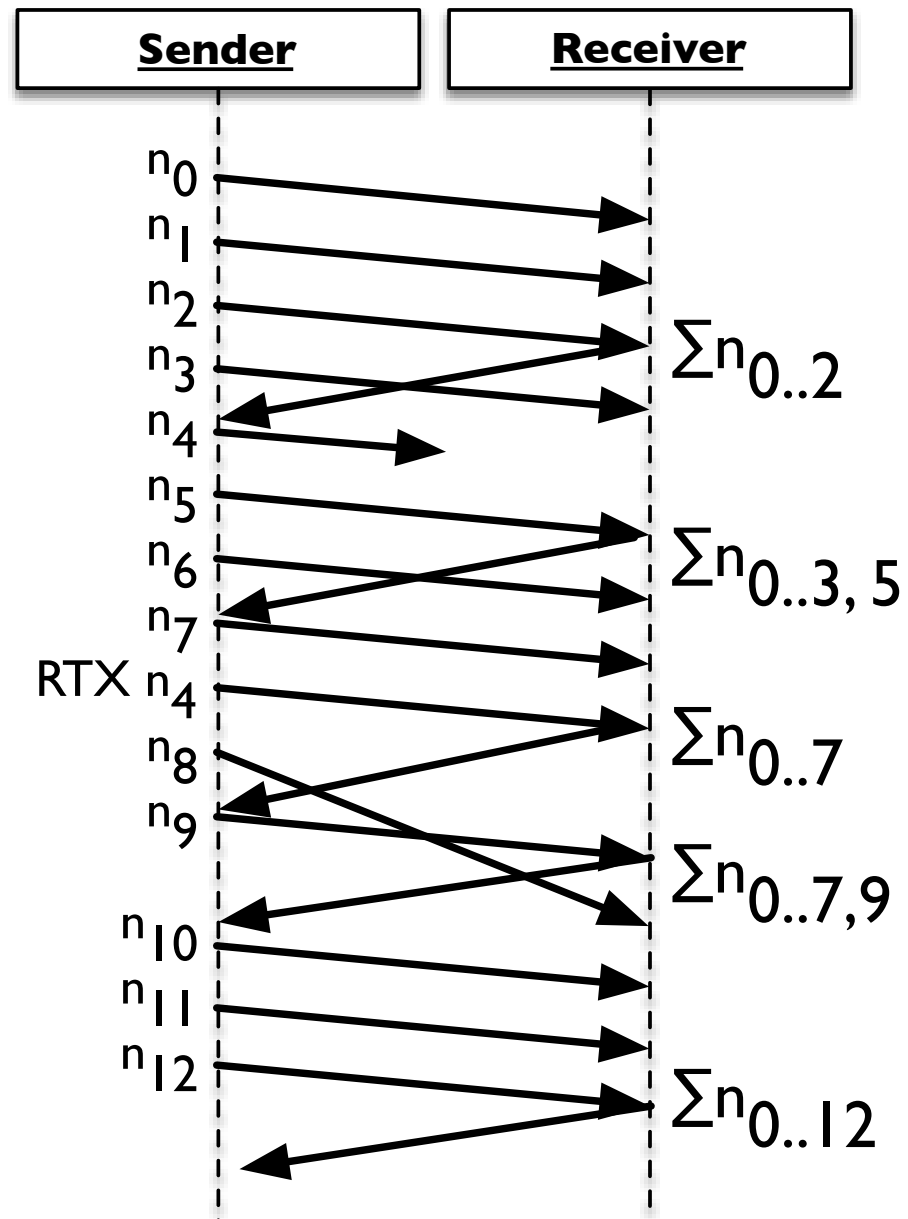- [9] Estan et al "Building a better NetFlow", SIGCOMM 2004.

Backup

Measurement Primitives Illustrated

Cumulative Nonce

Delta Timestamp Echo