# What if we'd designed measurement as a first-order service?

***Mirja Kühlewind*** and Brian Trammell, ETH Zürich

RIPE 72 - Copenhagen

May 24, 2016

measurement and architecture for a middleboxed internet

measurement | architecture | experimentation

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.*

*Supported by the Swiss State Secretariat for Education, Research and Innovation under contract number 15.0268. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.*

# Overview

- Network measurement is hard.
  - Which tool? What to measure? How often?
- Getting it right is even harder.
  - *„Wer misst, misst Mist“* *misst=measure & Mist=bullshit
- Why is it so hard?
  - “Big five” metrics (loss, latency, jitter, rate, reordering)
- How hard can it be?
  - Path layer providing explicit in-band measurement!

# Example: latency/RTT

- Ping?
  - IMCP often blocked
  - Differential Treatment possible
- 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

# Example: Loss/reordering

- TCP throughput testing… is hard to get right [1]
  - High network load and unwanted interference
- Ping Mesh?
  - Overhead is not applicable for Internet measurement
  - Do we really measure what we want to measure?
- TCP seq/ack number analysis for loss/reorder?
  - Always exposed, and roundly abused in the Internet
  - Only works with TCP

# 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.

- Passive measurement, e.g. Netflow/IPFIX:
  - Passive RTT measurement [2] broken by ACK optimizations [3], etc.
  - Inflexible, low-rate sampling, even though we know better [4].

# What do we really need?

- “Big five” metrics: loss, latency, jitter, rate, reordering
  - as socket properties, with API for access
  - exposed to the network, explicitly for measurability

- Transport-independent header fields explicitly defined for measurability
  - Constant-rate timestamps for latency/jitter
  - Exposure of loss/reordering
  - Detection of header manipulation (required for dynamic transport selection)
- Explicit endpoint control over measurement exposure
- Exposure in header allows passive as well as endpoint measurement

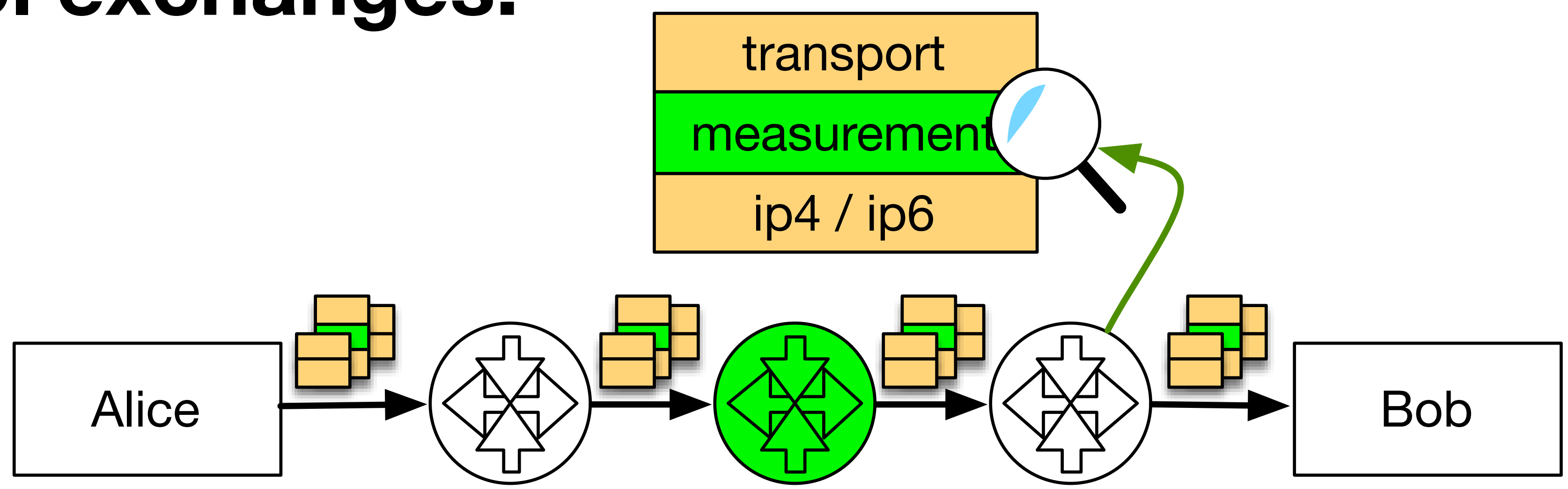
# Sounds great. Let's do it!

Now we just have to find the bits…

- IPv6 Destination Options [5]?
  - not very deployable, may be nearing deprecation, v6 only.
- IPv4 options?
  - even less deployable, v4 only.
- in the TCP header?
  - TCP only; options hard to deploy
  - HICCUPS [6] reclaimed a few bits from the header itself

# A Measurement Layer

… for explicit exposure of information as part of **normal protocol exchanges!**




➡ You don't have to instrument every packet, every endpoint, or every router to get *much* better information than we have today.

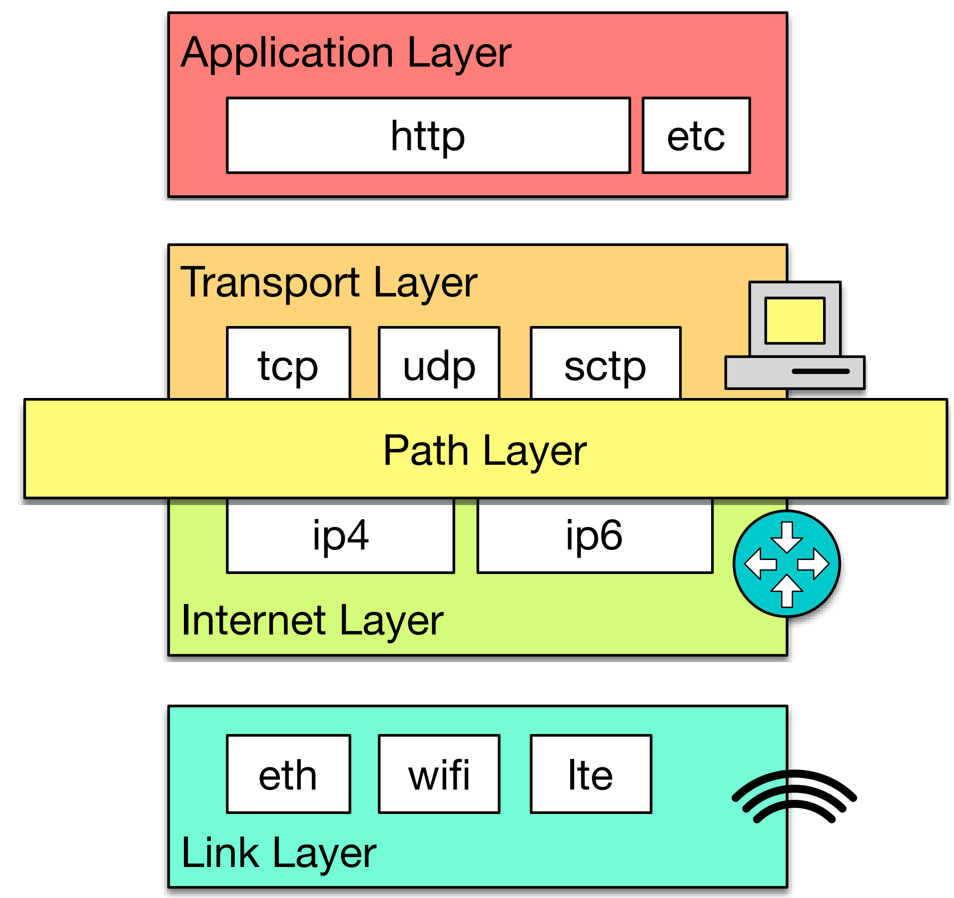
# Adding new layers to the stack for fun and profit

Our “measurement layer” is a special case of a more general problem [7]:

- Where do all of the complex, stateful, not necessarily end-to-end functions we’ve built go?

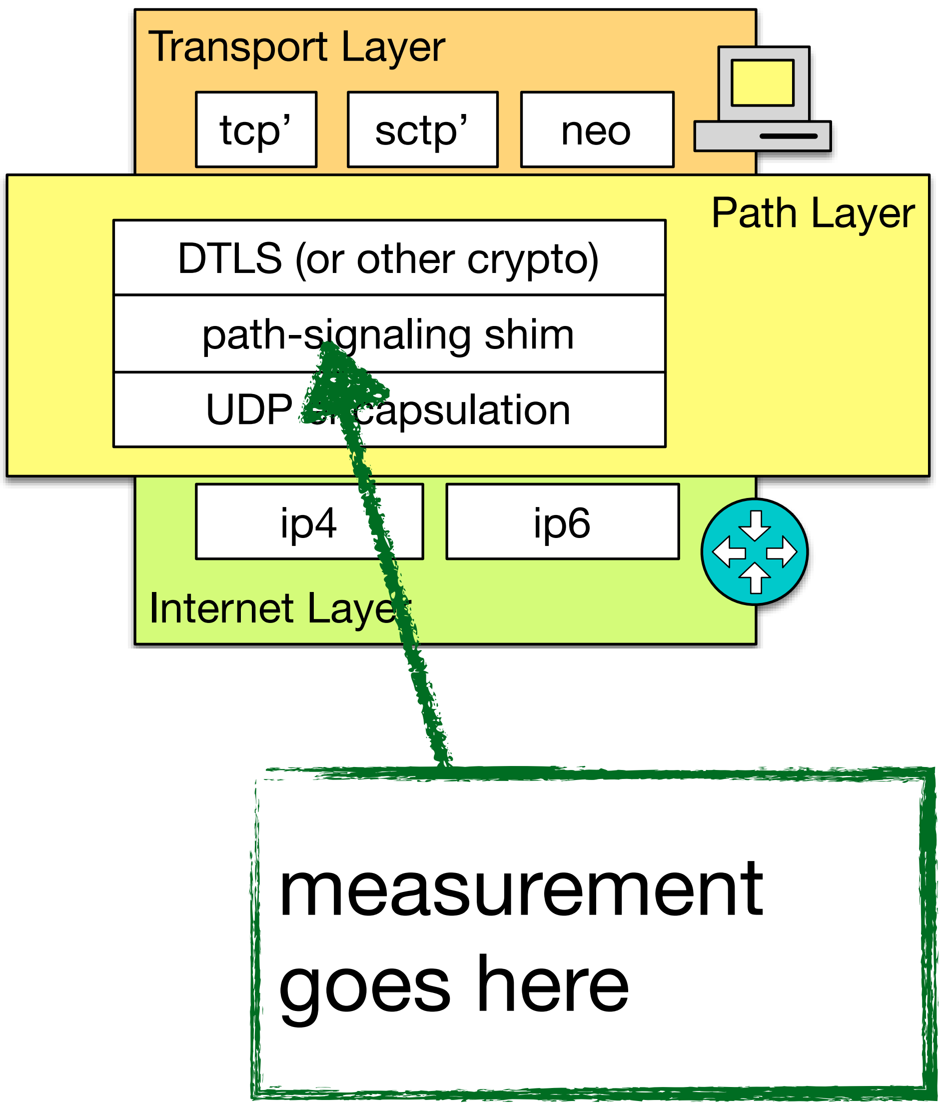
Solution: “**Path layer**”

- Encryption of transport layer and above to enforce end-to-end-ness
- Explicit exposure from endpoints to the path of appropriate information



# Path layer requirements

- 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 [8], spud@ietf.org



# Will it deploy?

- You can't add a new layer that today's routers won't route.
  - NAT: hard* to deploy protocols other than TCP or UDP

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.
- See presentation by Brian Trammell in MAT WG

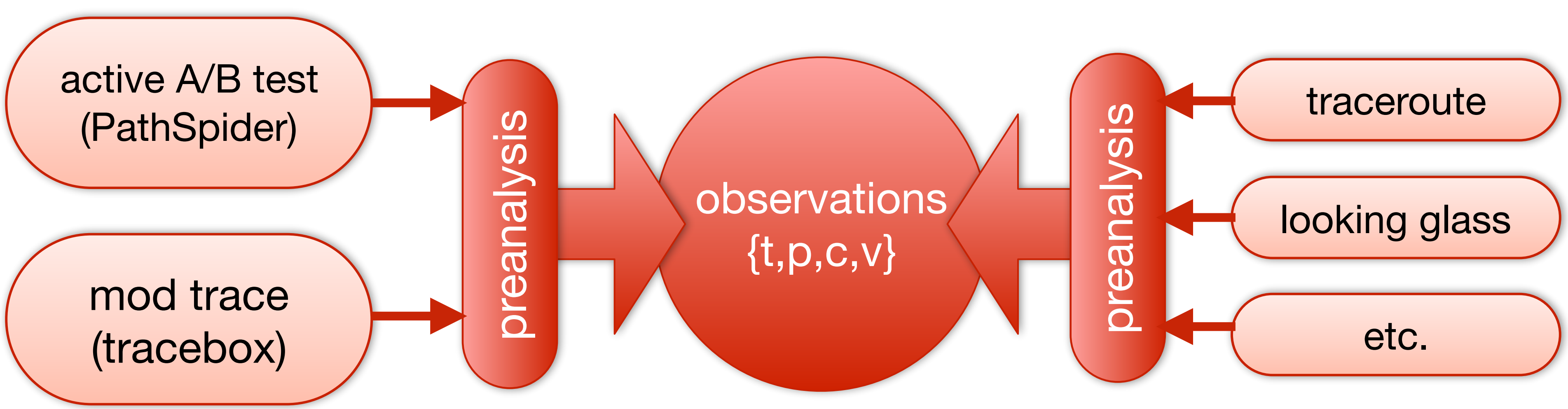
# Conculsion

- Yes, measurement is hard.
- Let's make it better!

**Missing:**
**Path layer for explicit exposure of traffic and measurement information**

# Path Transparency Observatory

- Observatory (public release end 2016) to derive common ***observations*** about *conditions* on a given *path* at a given *time*
  - Active measurements, made by the project
  - External measurements (e.g. traceroutes, BGP, traces)
- Combining disparate measurements leads to better insight
  - How likely is it that a certain path impairment impacts my traffic?




Follow http://mami-project.eu for updates on data model & availability!

# 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] Estan et al “Building a better NetFlow”, SIGCOMM 2004.

[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-kuehlewind-spud-use-cases (IETF individual Internet-draft)

[8] draft-trammell-spud-req (IETF individual Internet-draft)

# Backup

# The MAMI Project

## Measurement and Architecture for a Middleboxed Internet

**measurement**
of deployed middleboxes

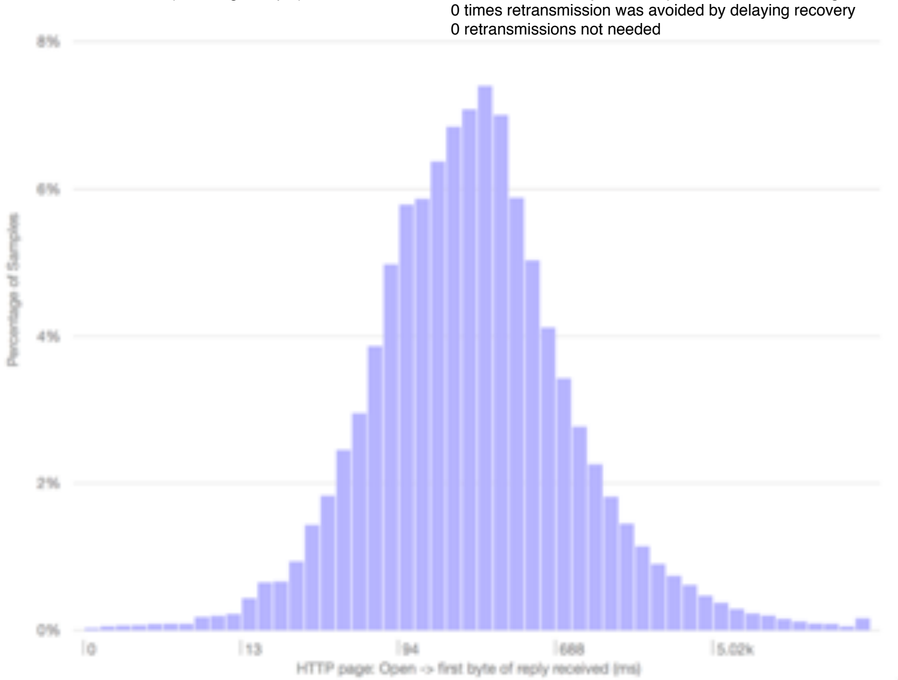
**architecture**
for middlebox cooperation

**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/**

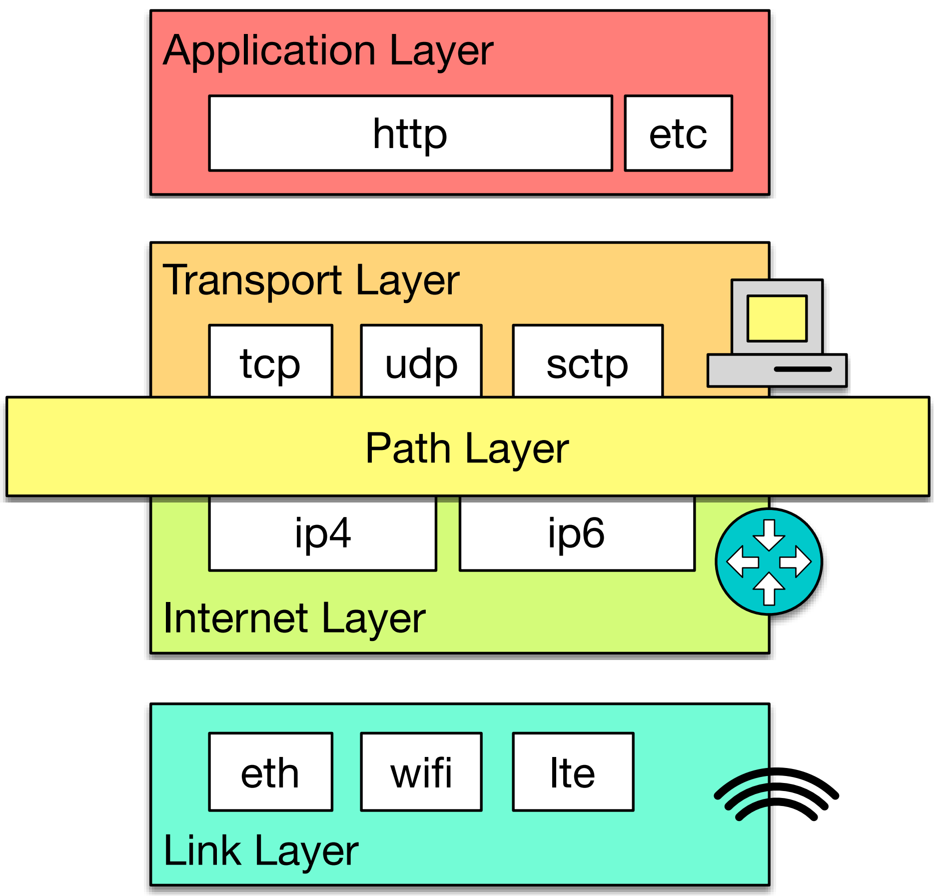
# How close are we to the goal?

```
% 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
```




- 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](telemetry.mozilla.org)
- Phase 1: generalizing and standardizing access to data we already have.
    - e.g. mPlane [4]

# Why a new shim layer?




- Transport layer: end-to-end sockets
  - flow information
  - stateful and ... g at the ...
- ... handling
  - ... rmation
  - ... and simple processing in the middle

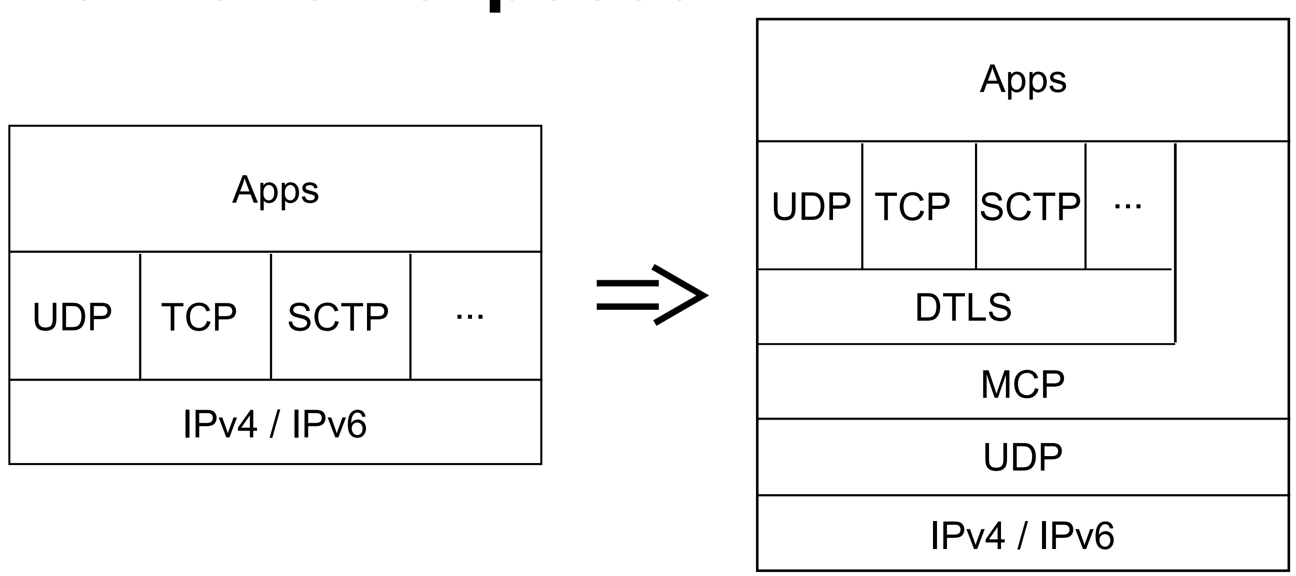
**Missing: Per-flow information for stateful in-network functions**

➡ **Path layer** for explicit cooperation with middleboxes instead of implicit assumptions

# Implementing an Explicit Path Interface

- Application can directly indicate requirements to path layer
- Transport can use the path layer to expose parts of its functionality/ intentions to the network
- *Middlebox Cooperation protocol* (MCP) signals these information appropriately to on-path middleboxes

➡ **Minimize the information exposed!**



# How to implement a new path layer?

- Transport-layer **encapsulation over UDP**
  - Need ports for NAT
  - Impossible to deploy with new protocol number across the Internet
  - Userspace (and kernelspace) implementation possible
- **Magic number** for easy recognition, protection against reflection
- **Flags** for “SYN/ACK” condition for state decision delegation to endpoint
  - All traffic bidirectional
  - Data in first packet possible
- Signals fit in a single packet (**no segmentation or reliability**)
- **Checksum** for error detection, cryptographic integrity checks available

# Why should I trust what you say about your flows?

- **Default**: *trust but verify*
  - declarative signaling: **no** negotiation, **no** guarantees
  - the best way to prevent cheating is to make it useless to do so
  - minimize the information exposed!

- Leverage existing trust relationships for higher-assurance declarations
  - e.g. your enterprise firewall, access network middleboxes, etc.

# 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?