TCP Options for Low Latency:

Maximum ACK Delay and

Microsecond Timestamps

Neal Cardwell
Yuchung Cheng
Eric Dumazet

Google

IETF 97: Seoul, Nov 2016

Motivation: lower latency, higher throughput

Datacenters with commodity 10Gbps Ethernet: RTT <100 us

Outdated fixed parameters:

- TCP Timestamps: granularity is 1 ms .. 1 sec [10x RTT]
- Delayed ACKs: typical delays: 40 ms .. 200 ms [400x RTT]
- RFC minimum RTO of 1 sec [10,000x RTT]

Goal: **negotiate** of values to fit today's networks

- Open-source Google's Linux TCP code for these
- Standardize option format and semantics in IETF

Minimum RTO

Most TCPs have min RTO of 200 ms .. 1 sec; why?

- Delayed ACKs: typical delays: 40 ms .. 200 ms [400x RTT]
- RFC minimum RTO of 1 sec [10,000x RTT]
  - But switches don't have 1 s. of buffer; hosts don't delay ACKs by 1 s.

Google experience, incast research [1] [2] [3]: lower timeouts help app latency

- Quicker RTO, TLP: simple way to vastly reduce latency ~40x (200ms -> 5ms)
- Google uses 5ms internally since 2013 ([3] mentions 5ms as well)

Maximum ACK Delay (MAD)

But if RTO is fast even when ACKs are delayed:

-> spurious retransmits and congestion control back-off

How to know long ACKs may be delayed?

Negotiate Max ACK Delay (MAD) in an option in TCP handshake...

Small MAD negotiated => enables small min RTO => improves performance

Microsecond TCP Timestamps: Motivation

TCP Timestamps [RFC1323][RFC7323]: finest allowed granularity is 1 ms

- But RTTs are < 100 us and soon 10 us in the datacenter

Benefits of 1 us TCP timestamps:

- Can undo cwnd reductions in datacenter
  - In datacenter, original and fast retransmit have same 1ms timestamp
  - Can't use TCP timestamps to undo cwnd reduction [RFC3522/RFC4015]
- Can do fine-grained measurement and diagnostics
  - One-way delay variation for data (e.g. incast queues), ACKs

Microsecond TCP Timestamps: Implementation

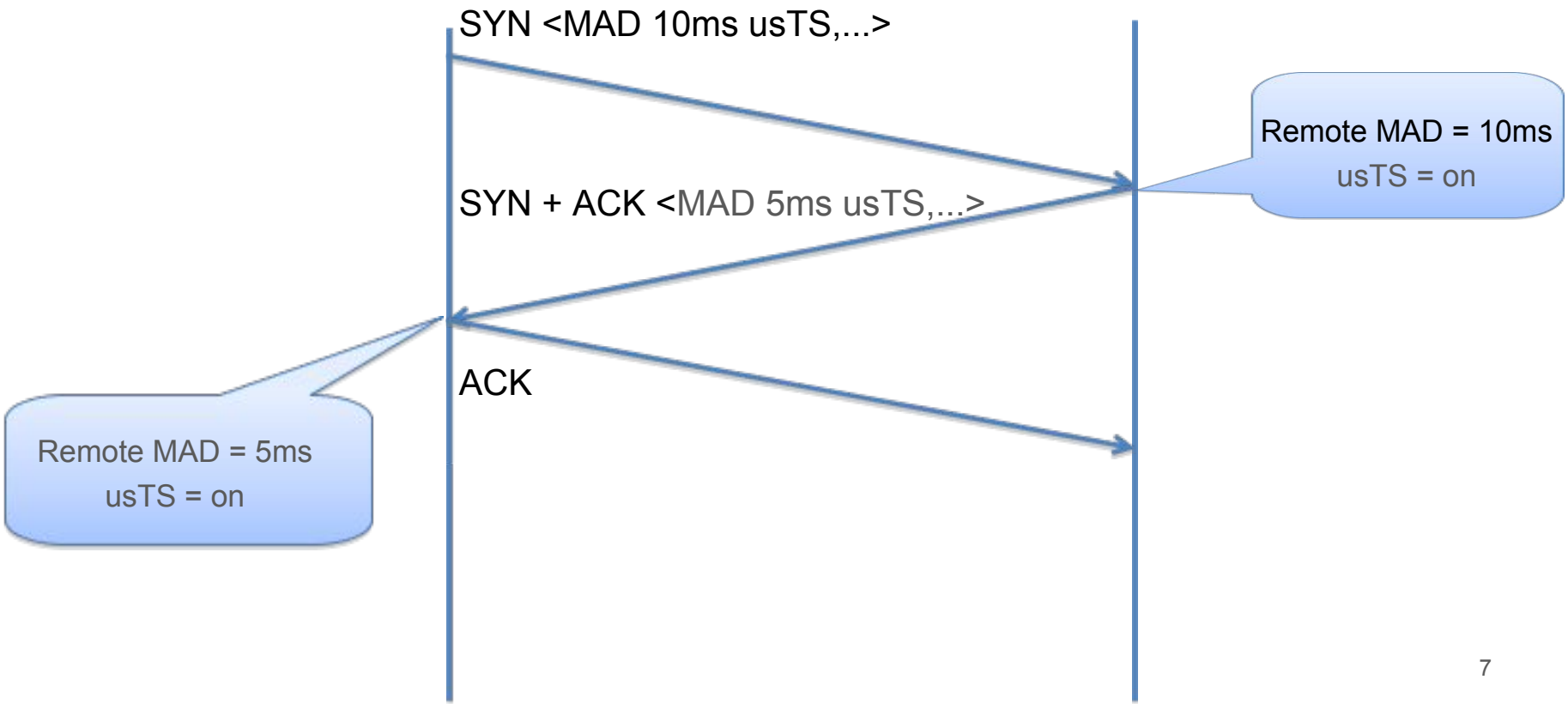
When using usec TS, need to adjust a constant in PAWS logic

- When to expect 32-bit wrap-around in idle connections [RFC7323 sec 5.5]:
  - 1 ms => wraps in ~24 days
  - 1 us => wraps in ~34 minutes
  - 1 ns => wraps in ~2 secs

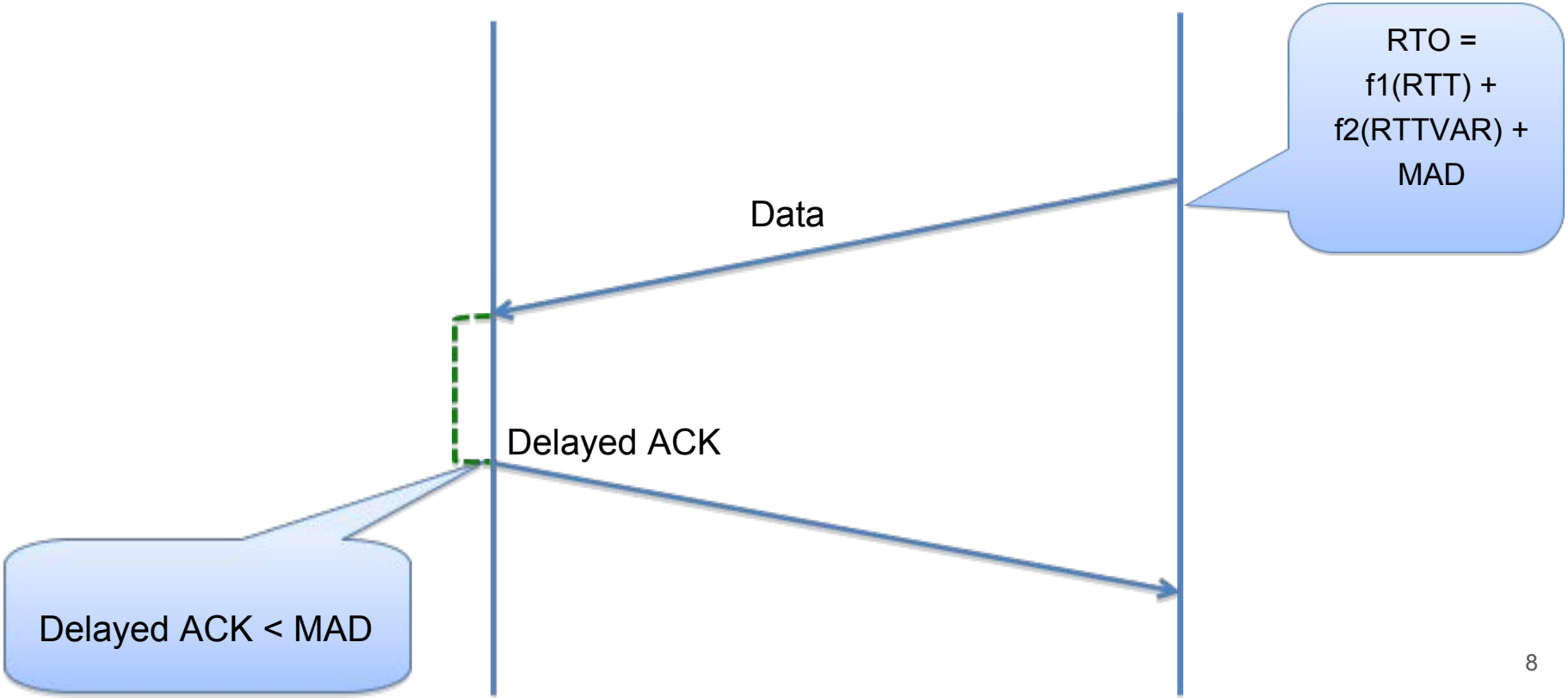
How? **Negotiate use in an option** in TCP handshake...

- Handles the general/cloud/SaaS case
- (Could also use per-route config if this is intradomain traffic)

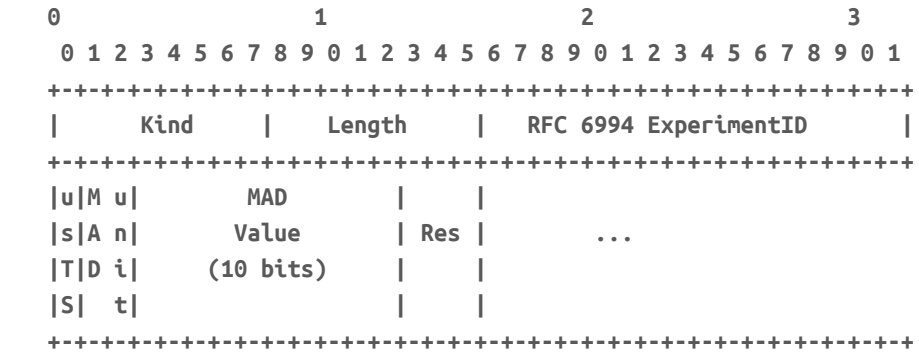
Options and the TCP Handshake



Max ACK Delay and Min RTO interaction



Low Latency Option: Proposed Format



usTS: use microsecond timestamps? (0 = no usec Timestamps; 1 = usec Timestamps)

MAD unit: time units for MAD value (0 = no MAD negotiated; 1=msecs, 2=usecs, 3=nsecs)

MAD value: Maximum ACK Delay value (1 ... 999)

Total space:

- 6 bytes - using 2-byte RFC 6994 ExperimentID
- 4 bytes - if promoted to standard with its own option Kind (no ExperimentID)

Status

Code for these 2 features has matured at Google (used for all internal TCP traffic)

- Maximum ACK Delay: since Jul 2005 (yes, 11 years ago!)
- Microsecond timestamps: since Feb 2015 (1.75 years ago)

Verbal interest from at least one other major TCP implementor for MAD
Next steps:

1. Internet Draft
2. Change code to support RFC6994 experimental option format
3. Send code upstream to Linux by Q1 2017