Tail Loss Probe (TLP)

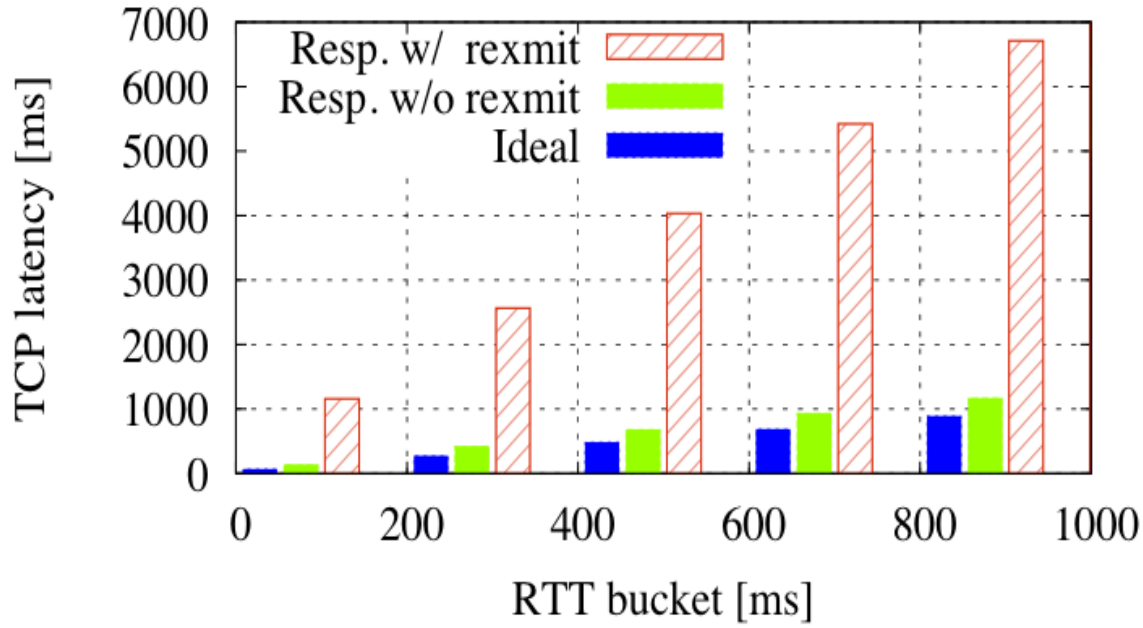
Converting RTOs to fast recoveries

draft-dukkipati-tcpm-tcp-loss-probe-00

Nandita Dukkipati, Neal Cardwell,
Yuchung Cheng, Matt Mathis
{nanditad, ncardwell, ycheng, mattmathis}@google.com

Losses hurt Web latency

- Lossy responses last 10 times longer than lossless ones.
- 6.1% responses and 30% of TCP connections experience losses.



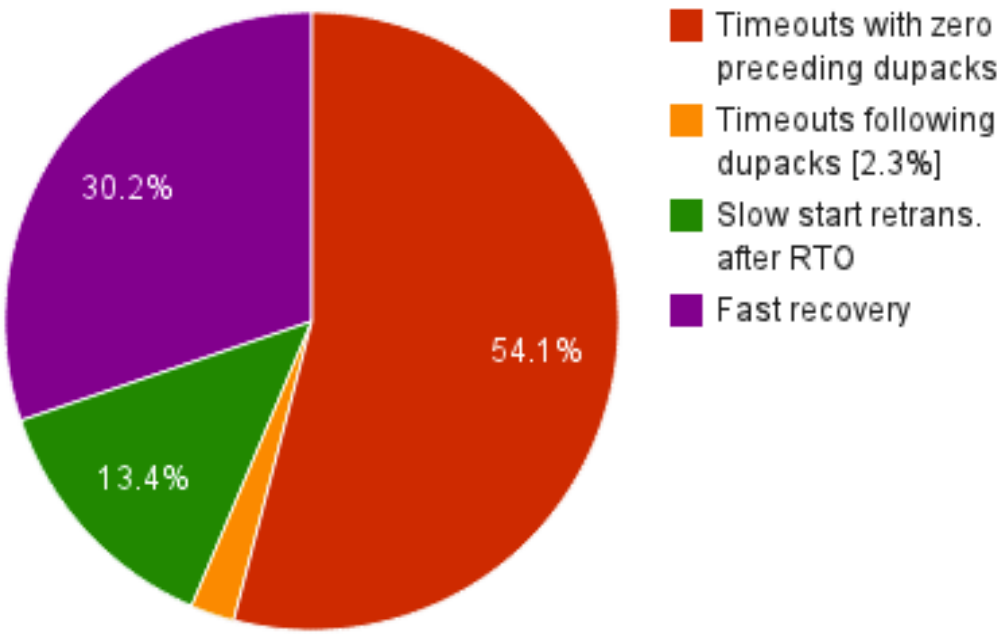
- Problem: timeouts are expensive for short flows
  - RTO is primary recovery mode for Web traffic
  - Normalized RTO values (#RTTs)

50%ile	75%ile	90%ile	95%ile	99%ile
5	12	29	54	214

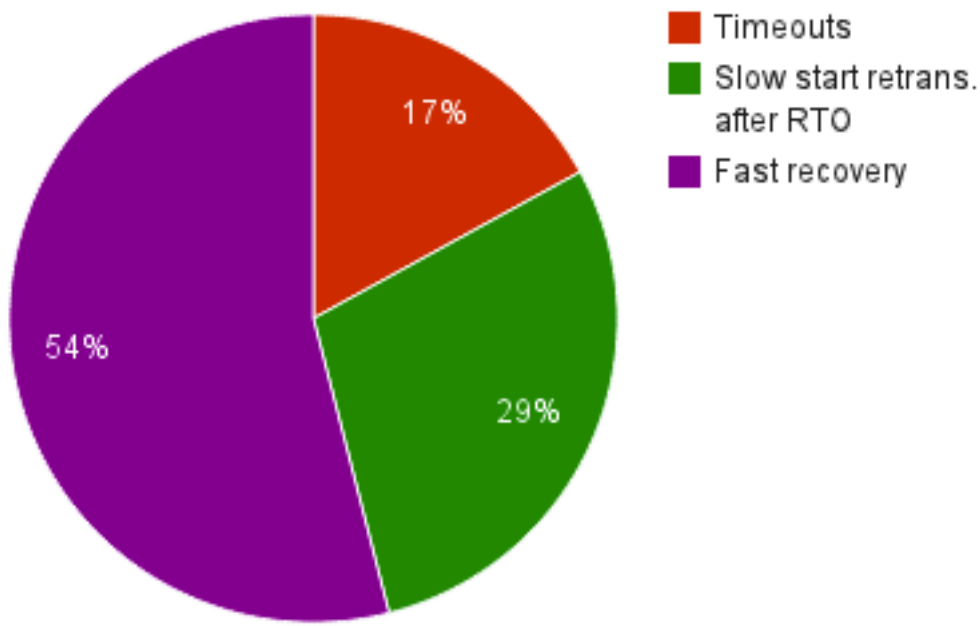
How does TCP recover from losses?

TCP retransmission breakdown in two Google DCs.

Web



YouTube

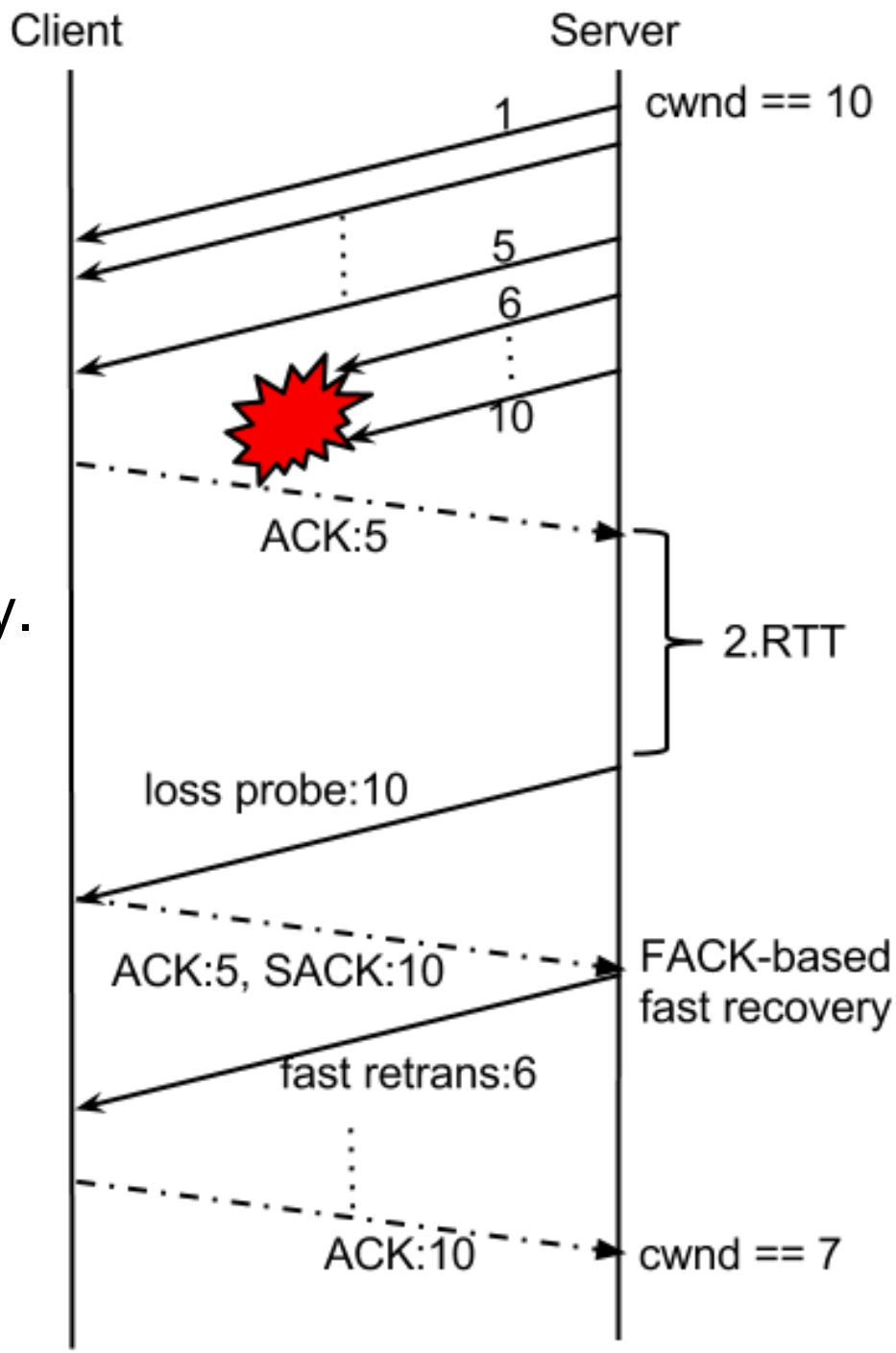


- Tail segments are twice more likely to be lost than start ones.
- Losses are bursty and contiguous. [A L *] pattern more common than [A L * S * L].

Tail Loss Probe (TLP)

Key idea: convert RTOs to fast recovery.

- Transmit loss probe after approx. 2. RTT in absence of ACKs.
- Retransmit last packet (or new if available) to trigger fast recovery.



TLP example

TLP pseudocode

Probe timeout (PTO): timer event indicating that an ACK is overdue.

Schedule probe on transmission of new data in Open state:

```
    -> Either cwnd limited or application limited.
    -> RTO is farther than PTO.
    -> FlightSize > 1: schedule PTO in max(2*SRTT, 10ms).
    -> FlightSize == 1: PTO is max(2*SRTT, 1.5*SRTT+WCDelAckT)
```

When probe timer fires:

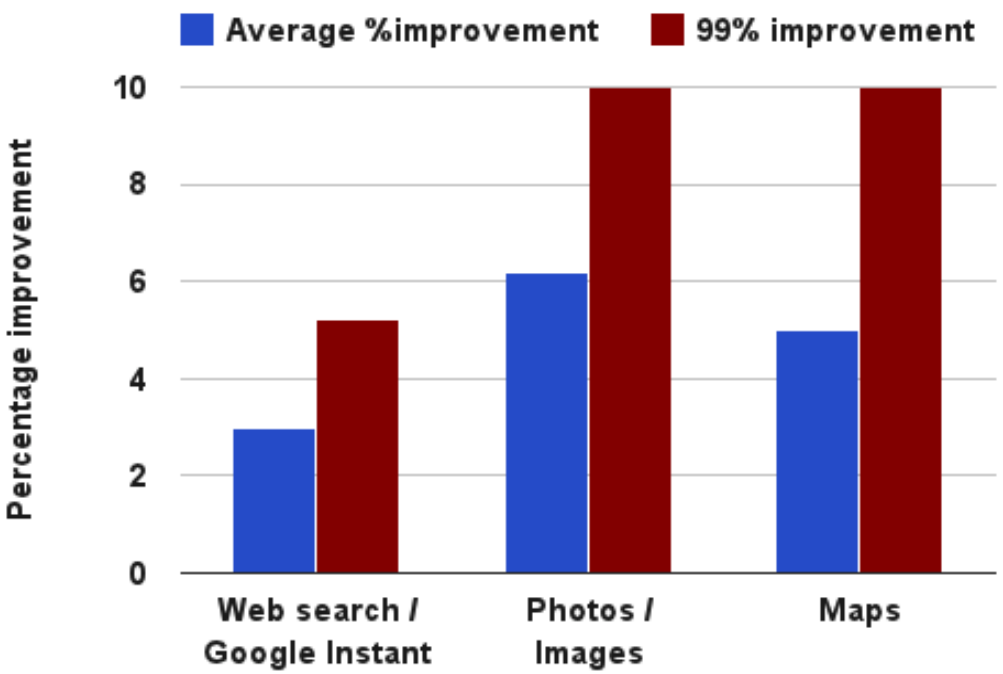
```
(a) If a new previously unsent segment exists:
        -> Transmit new segment.
        -> FlightSize += SMSS. cwnd remains unchanged.

(b) If no new segment exists:
        -> Retransmit the last segment.

(c) Reschedule PTO.
```

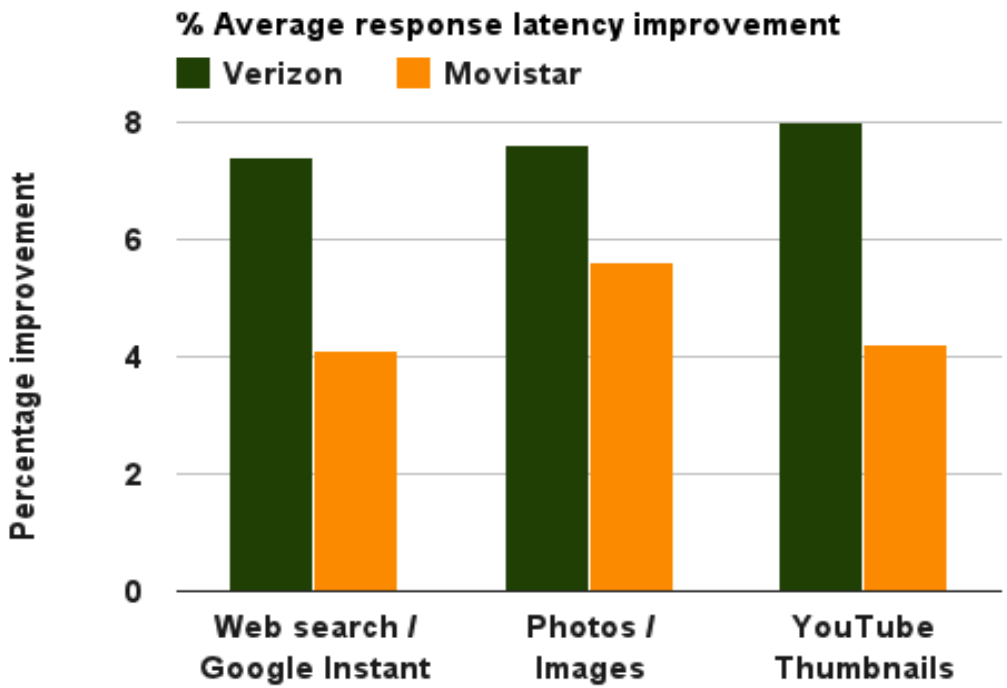
ACK processing:

```
    -> Cancel any existing PTO.
    -> Reschedule PTO relative to time at which the ACK is received
```

Experiments with TLP

- 2-way experiment over 10 days: Linux baseline versus TLP.
- 6% avg. reduction in HTTP response latency for image search.
- 10% reduction in RTO retransmissions.
- 0.6% probe overhead.





Mobile only

Detecting repaired losses: basic algorithm

- Problem: congestion control not invoked if TLP repairs loss **and** the only loss is last segment.
- Basic idea
  - TLP episode: N consecutive TLP segments for same tail loss.
  - End of TLP episode: ACK above SND.NXT.
  - Expect to receive N TLP dupacks before episode ends
- Algorithm is conservative: cwnd reduction can occur with no loss.
  - Delayed ACK timer.
  - ACK loss.

TLP properties

- Property 1: Unifying recovery regardless of loss position.
  - Example: 10 packet burst. Last or middle segment losses are both recovered via fast recovery.
- Property 2: fast recovery of any N-degree tail loss for any sized transaction.
  - TLP combined with Early-retransmit variant recovers any tail loss via fast recovery.

TLP properties (contd.)

#losses	scoreboard after TLP ACKed	mechanism	outcome
A A A **L**	A A A A	TLP loss detection	All repaired
A A **L L**	A A **L S**	Early retransmit	All repaired
A **L L L**	A **L L S**	Early retransmit	All repaired
L L L L	**L L L S**	FACK fast recovery	All repaired
>=5 L	**...L S**	FACK fast recovery	All repaired

Key:
A = ACKed; **L** = Lost; **S** = SACKed segment.

Conclusion

- Bursty applications have made end of transaction losses a common case.
- TLP unifies TCP's loss recovery schemes by allowing fast recovery of any N-degree tail loss.
- Simple to implement and deploy.
- What's next? Forward Error Correction (FEC) in TCP.