# Maelstrom: Transparent Error Correction for Lambda Networks

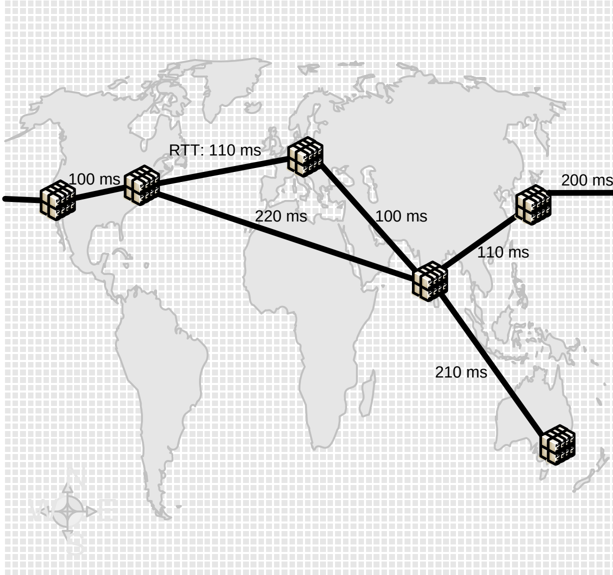
*Mahesh Balakrishnan*, Tudor Marian, Ken Birman
Hakim Weatherspoon, Einar Vollset

Cornell University

# Lambda Networks

Bandwidth is ubiquitous.

Why can't we use it?

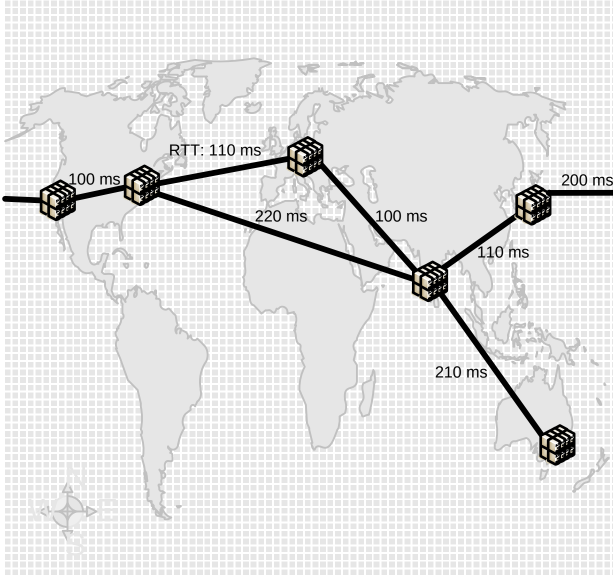


# Lambda Networks

Bandwidth is ubiquitous.

Why can't we use it?

TCP collapses!



# Reliable Communication between Datacenters

TCP fails in three ways:

1. *Throughput Collapse*
   100ms RTT, 0.1% Loss, 40 Gbps $\rightarrow$ Tput $<$ 10 Mbps!
2. *Massive Buffers* required for High-Rate Traffic
3. *Recovery Delays* for Time-Critical Traffic

# Reliable Communication between Datacenters

TCP fails in three ways:

1. *Throughput Collapse*
   100ms RTT, 0.1% Loss, 40 Gbps → Tput $<$ 10 Mbps!
2. *Massive Buffers* required for High-Rate Traffic
3. *Recovery Delays* for Time-Critical Traffic

Current Solutions:

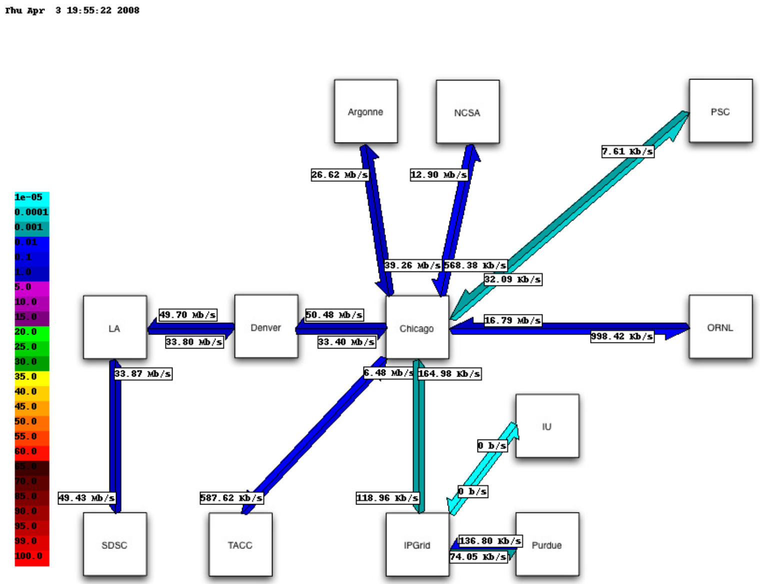
- One Flow → Multiple Split Flows
- Resize Buffers
- *Spend (infinite) money!*

# Is Perfection Achievable?

Is Perfection ~~Achievable~~?

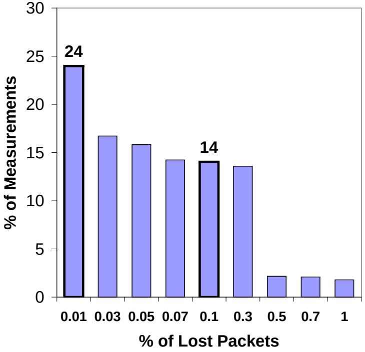
Achieved

# TeraGrid: Supercomputer Network



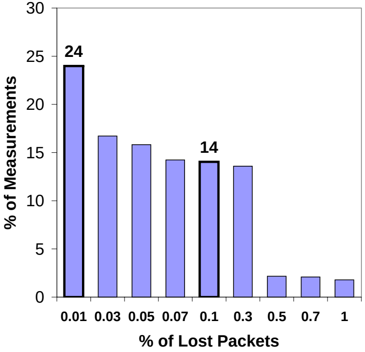
# TeraGrid: Supercomputer Network

- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
  - transient congestion
  - degraded fiber
  - malfunctioning HW
  - misconfigured HW
  - switching contention
  - low receiver power
  - end-host overflow
  - ...



# TeraGrid: Supercomputer Network

- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
  - transient congestion
  - degraded fiber
  - malfunctioning HW
  - misconfigured HW
  - switching contention
  - low receiver power
  - end-host overflow
  - ...



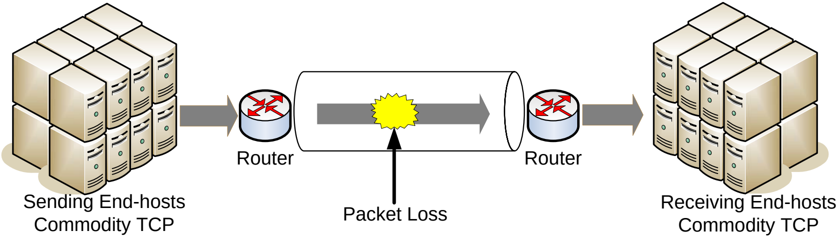

Electronics: Cluttered Pathways

Optics: Lossy Fiber

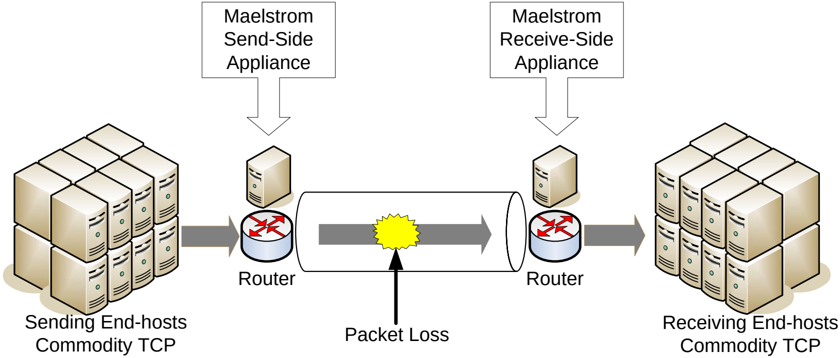
# Problem Statement

Run unmodified TCP/IP over lossy high-speed long-distance networks

# The Maelstrom Network Appliance

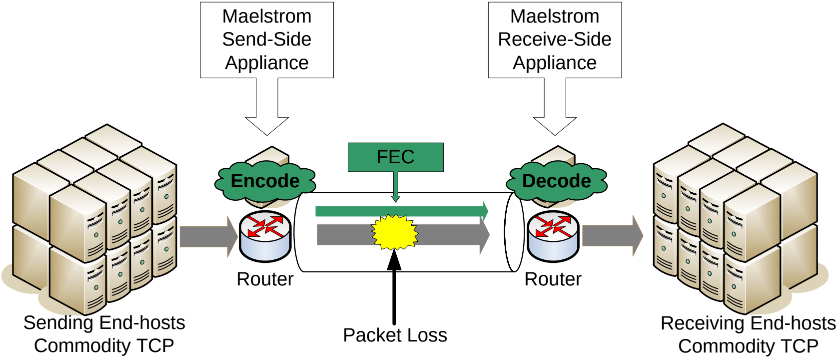


# The Maelstrom Network Appliance




Transparent: No modification to end-host or network

# The Maelstrom Network Appliance




Transparent: No modification to end-host or network

FEC = Forward Error Correction

# What is FEC?


3 repair packets from every 5 data packets

Receiver can recover from any 3 lost packets

Rate[1]: $(r, c)$ — $c$ repair packets for every $r$ data packets.

- Pro: Recovery Latency independent of RTT
- Constant Data Overhead: $\frac{c}{r+c}$
- Packet-level FEC at End-hosts: Inexpensive, No extra HW

[1] Rateless codes are popular, but inapplicable to real-time streams

# What is FEC?


3 repair packets from every 5 data packets

Receiver can recover from any 3 lost packets

Rate[1]: $(r, c)$ — $c$ repair packets for every $r$ data packets.

- Pro: Recovery Latency independent of RTT
- Constant Data Overhead: $\frac{c}{r+c}$
- Packet-level FEC at End-hosts: Inexpensive, No extra HW
- Con: Recovery Latency dependent on channel data rate

[1] Rateless codes are popular, but inapplicable to real-time streams

# What is FEC?


3 repair packets from every 5 data packets

Receiver can recover from any 3 lost packets

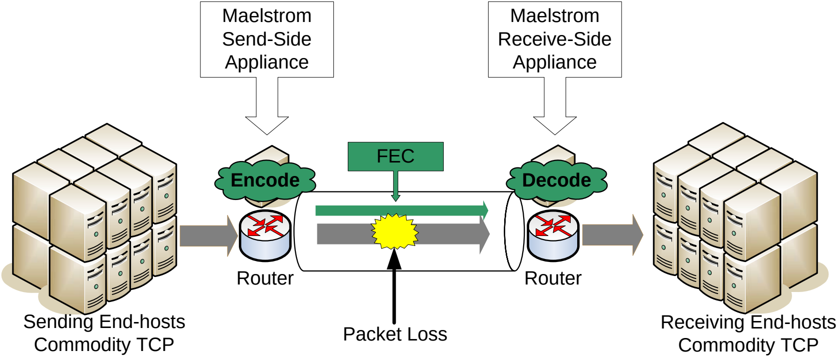
Rate[1]: $(r, c)$ — $c$ repair packets for every $r$ data packets.

- Pro: Recovery Latency independent of RTT
- Constant Data Overhead: $\frac{c}{r+c}$
- Packet-level FEC at End-hosts: Inexpensive, No extra HW
- Con: Recovery Latency dependent on channel data rate

Solution: End-to-End FEC between Datacenters

[1] Rateless codes are popular, but inapplicable to real-time streams

# The Maelstrom Network Appliance




Transparent: No modification to end-host or network

FEC = Forward Error Correction

Where: at the appliance, What: aggregated data

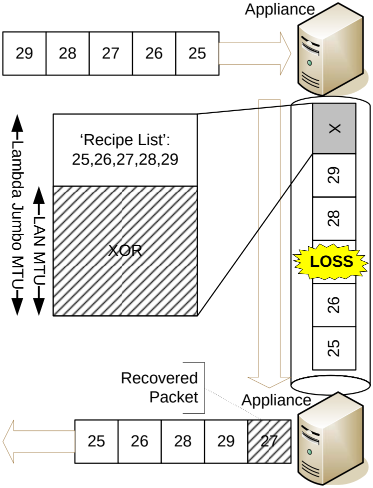
# Maelstrom Mechanism

Send-Side Appliance:

- Snoop IP packets
- Create repair packet = XOR + 'recipe' of data packet IDs

Receive-Side Appliance:

- Lost packet recovered using XOR and other data packets
- At receiver end-host: out of order, no loss



# FEC and Bursty Loss

- $(r, c)$ code can tolerate burst of $c$ losses
- Existing solution: *interleaving*
- Interleave $i$ and rate $(r, c)$ tolerates $(c * i)$ burst...
- ...with $i$ times the latency

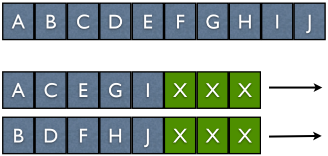



Figure: Interleave of 2 — Even and Odd packets encoded separately

Current: Recovery Latency $\propto$ maximum burst size

# FEC and Bursty Loss

- $(r, c)$ code can tolerate burst of $c$ losses
- Existing solution: *interleaving*
- Interleave $i$ and rate $(r, c)$ tolerates $(c * i)$ burst...
- ...with $i$ times the latency

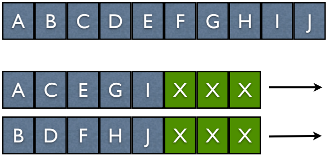
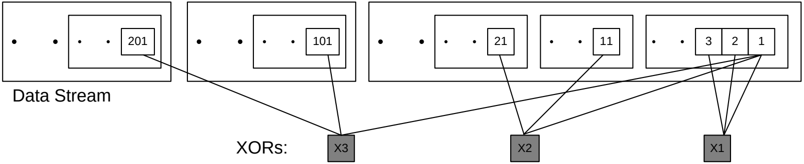



Figure: Interleave of 2 — Even and Odd packets encoded separately

Current: Recovery Latency $\propto$ maximum burst size
Wanted: Recovery Latency $\propto$ actual burst size

# Layered Interleaving for Bursty Loss

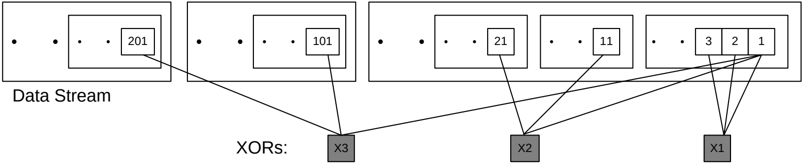
Recovery Latency $\propto$ Actual Burst Size, not Max Burst Size



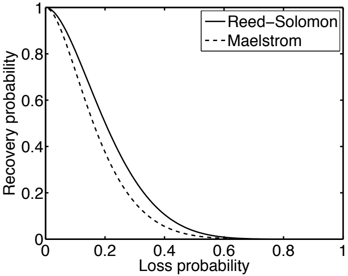

- XORs at different interleaves
- Recovery latency degrades gracefully with loss burstiness:
  X1 catches random singleton losses
  X2 catches loss bursts of 10 or less
  X3 catches bursts of 100 or less

# Layered Interleaving for Bursty Loss

Recovery Latency $\propto$ Actual Burst Size, not Max Burst Size



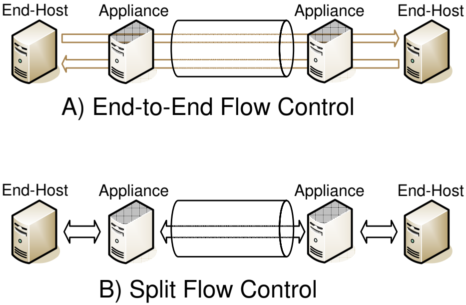

- XORs at different interleaves
- Recovery latency degrades gracefully with loss burstiness:
  X1 catches random singleton losses
  X2 catches loss bursts of 10 or less
  X3 catches bursts of 100 or less




Comparison of Recovery Probability: r=7, c=2

# TCP Traffic — Flow Control

- Two Flow Control Modes for TCP/IP Traffic:




- Split Mode avoids client buffer resizing (PeP)

# UDP Traffic

- Works for UDP-based protocols
  - Reliable multicast
  - High-speed data transfer
  - VoIP, video streaming, etc.
- What about loss at end-host? (kernel overflow, bad NIC)
- Maelstrom receive-side proxy acts as a *packet cache*:
  - Requires compatible protocol design
  - Or knowledge of protocol internals

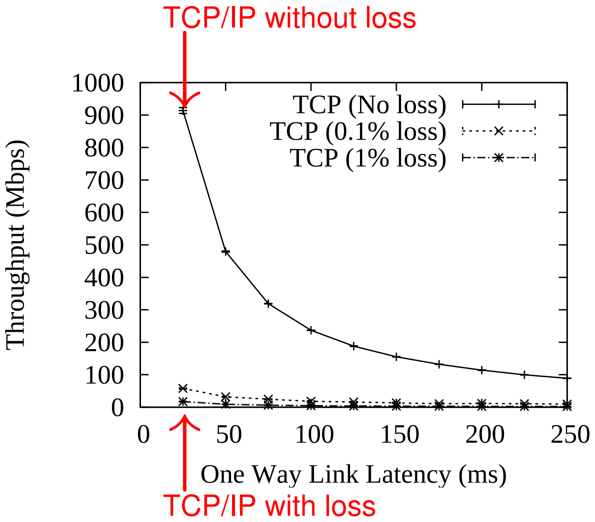
# Implementation Details

- In Kernel — Linux 2.6.20 Module
- Commodity Box: 3 Ghz, 1 Gbps NIC ($\approx$ 800$)
- Max speed: 1 Gbps, Memory Footprint: 10 MB
- 50-60% CPU $\rightarrow$ NIC is the bottleneck (for $c = 3$)

- How do we efficiently store/access/clean a gigabit of data every second?
- Scaling to Multi-Gigabit: Partition IP space across proxies

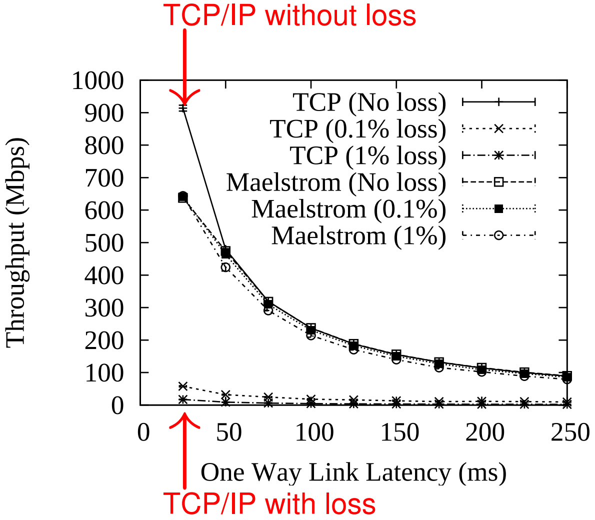
# Evaluation: FEC Mode and Loss

Claim: Maelstrom solves Problem #1 with TCP (Throughput Collapse)



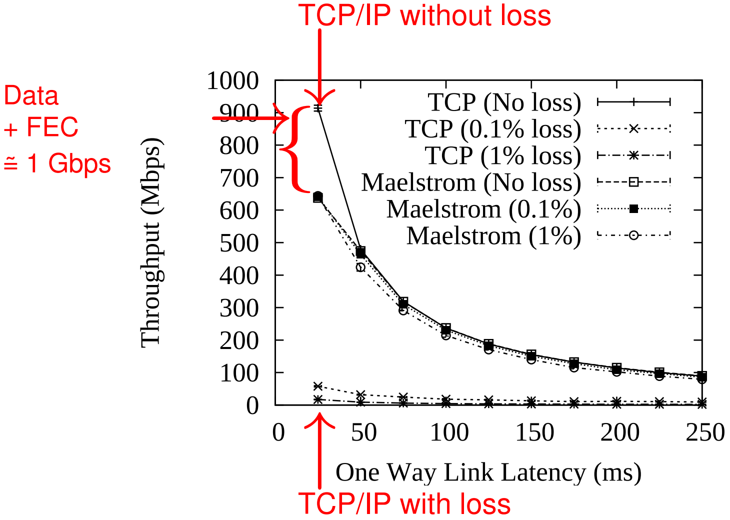
# Evaluation: FEC Mode and Loss

Claim: Maelstrom solves Problem #1 with TCP (Throughput Collapse)



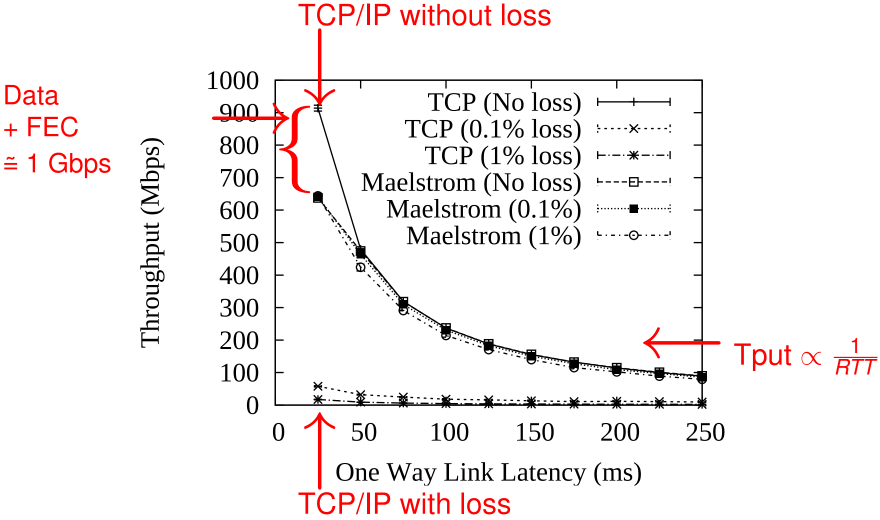
# Evaluation: FEC Mode and Loss

Claim: Maelstrom solves Problem #1 with TCP (Throughput Collapse)



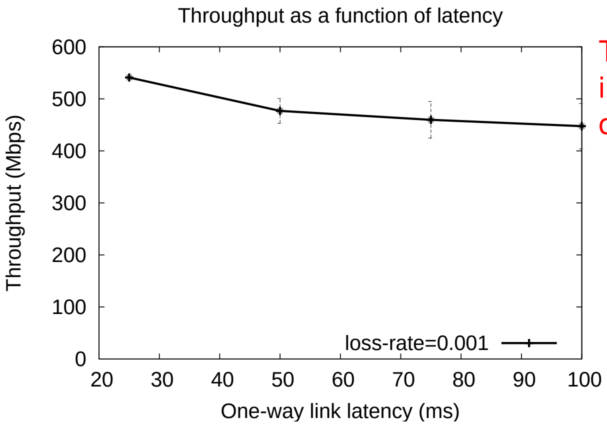
# Evaluation: FEC Mode and Loss

## Claim: Maelstrom solves Problem #1 with TCP (Throughput Collapse)



# Evaluation: Split Mode and Buffering

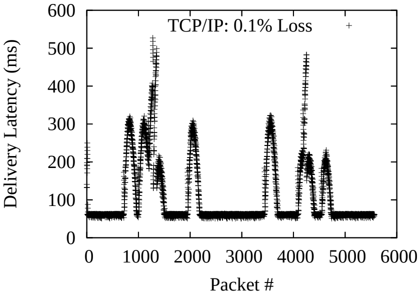
## Claim: Maelstrom solves Problem #2 with TCP (Massive Buffer Requirement)

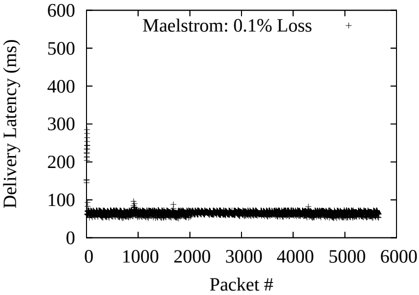



Tput independent of RTT!

# Evaluation: Delivery Latency

## Claim: Maelstrom solves Problem #3 with TCP (Recovery Delays)



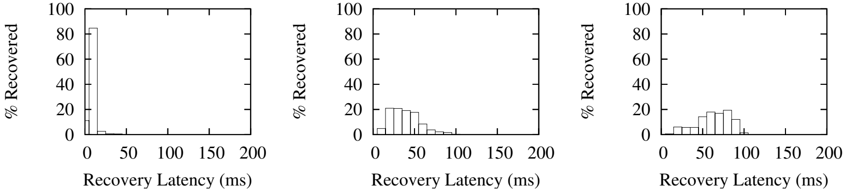

Sources of Jitter:

- Receive-side buffering due to sequencing
- Send-side buffering due to congestion control

# Evaluation: Layered Interleaving

## Claim: Recovery Latency depends on Actual Burst Length

Burst Length = 1 20 40




- Longer Burst Lengths → Longer Recovery Latency

# Conclusion

- Problems with TCP on Lambda Networks:
  - Throughput Collapse due to Sensitive Congestion Control
  - Massive Buffers required at End-Hosts
  - Recovery Delays due to RTT dependence
- The Maelstrom Appliance:
  - End-to-End FEC between Datacenters
  - Completely Transparent

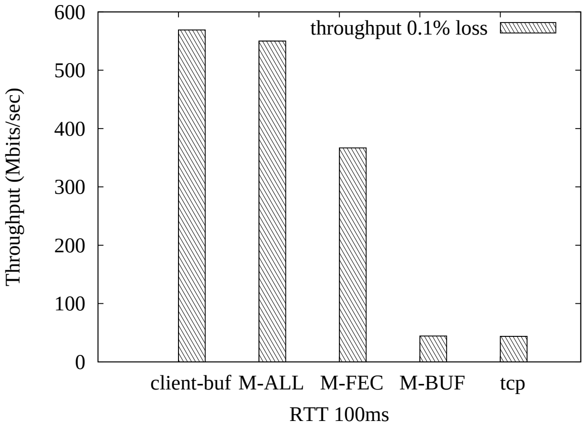
# Conclusion

- Problems with TCP on Lambda Networks:
  - Throughput Collapse due to Sensitive Congestion Control
  - Massive Buffers required at End-Hosts
  - Recovery Delays due to RTT dependence
- The Maelstrom Appliance:
  - End-to-End FEC between Datacenters
  - Completely Transparent

Thank You!

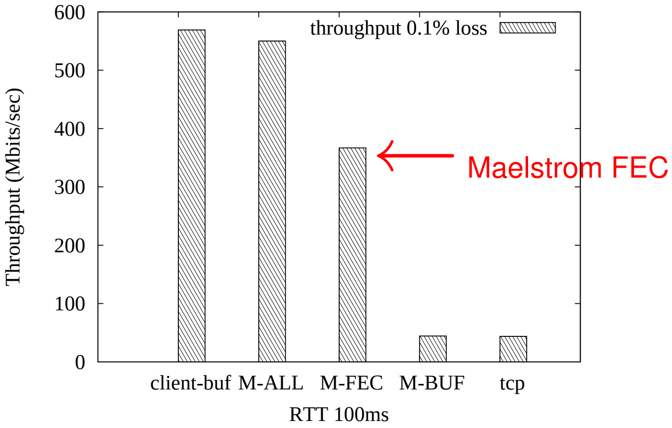
# Extra Slide: Split Mode and Buffering

Claim: Maelstrom eliminates the need for buffer tuning



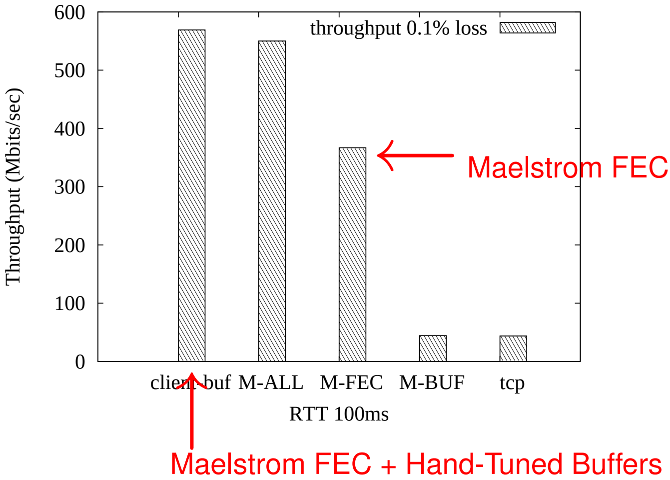
# Extra Slide: Split Mode and Buffering

Claim: Maelstrom eliminates the need for buffer tuning



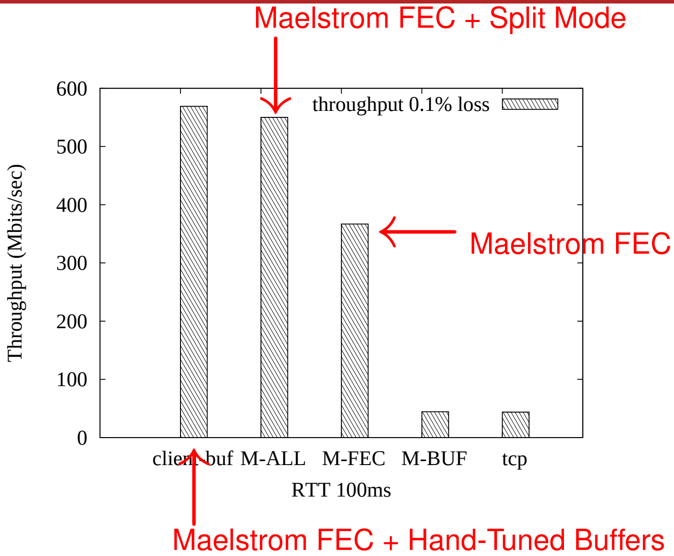
# Extra Slide: Split Mode and Buffering

Claim: Maelstrom eliminates the need for buffer tuning



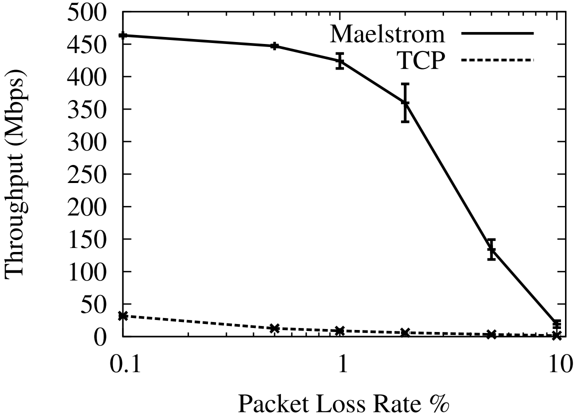
# Extra Slide: Split Mode and Buffering

Claim: Maelstrom eliminates the need for buffer tuning



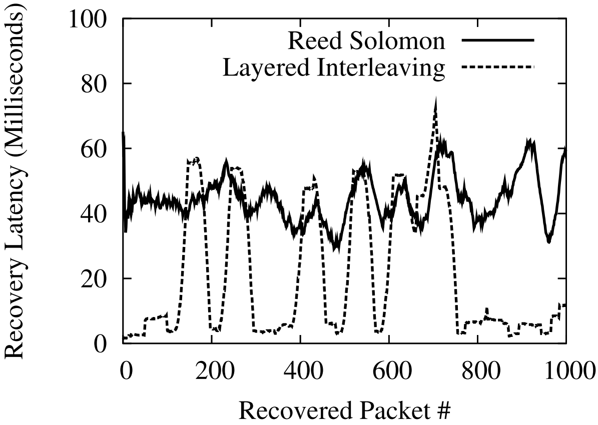
# Extra Slide: FEC mode and loss

Claim: Maelstrom works at high loss rates



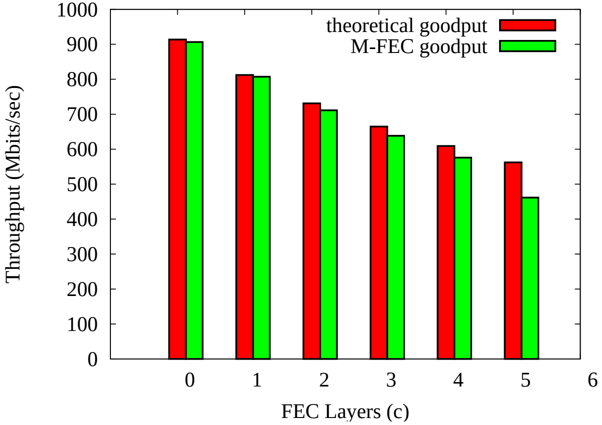

Link RTT = 100ms

# Extra Slide: Layered Interleaving



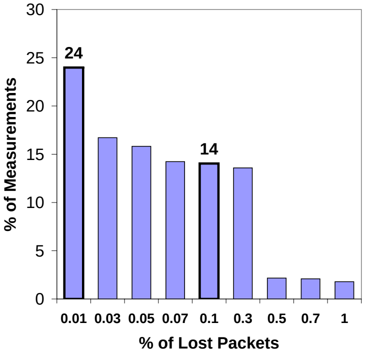
# Extra Slide: Maelstrom Evaluation

## Maelstrom goodput is near theoretical maximum



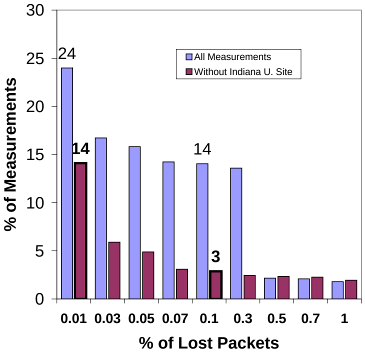
# Extra Slide: TeraGrid: Supercomputer Network

- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
    - transient congestion
    - degraded fiber
    - malfunctioning HW
    - misconfigured HW
    - switching contention
    - low receiver power
    - end-host overflow
    - ...



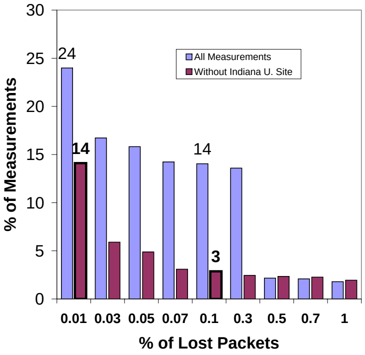
# Extra Slide: TeraGrid: Supercomputer Network

- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
  - transient congestion
  - degraded fiber
  - malfunctioning HW
  - misconfigured HW
  - switching contention
  - low receiver power
  - end-host overflow
  - ...



# Extra Slide: TeraGrid: Supercomputer Network

- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
  - transient congestion
  - degraded fiber
  - malfunctioning HW
  - misconfigured HW
  - switching contention
  - low receiver power
  - end-host overflow
  - ...




Problem Statement: Run *unmodified* TCP/IP over *lossy* high-speed long-distance networks