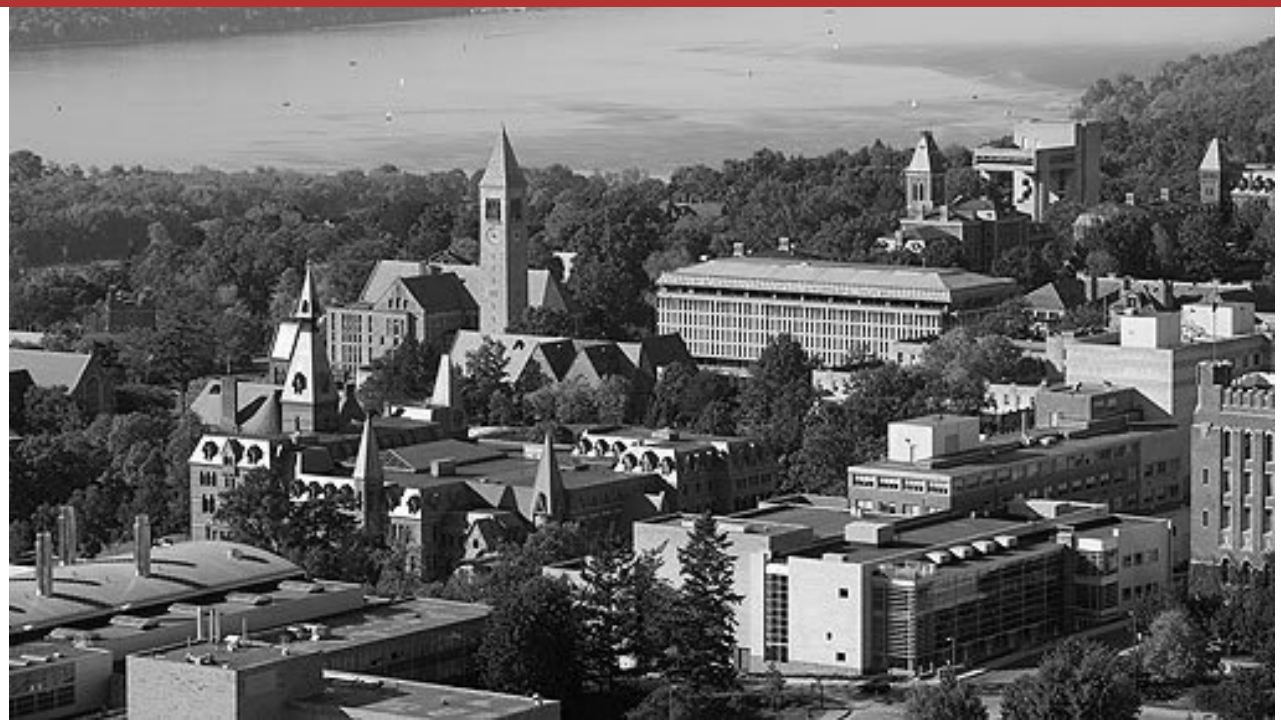
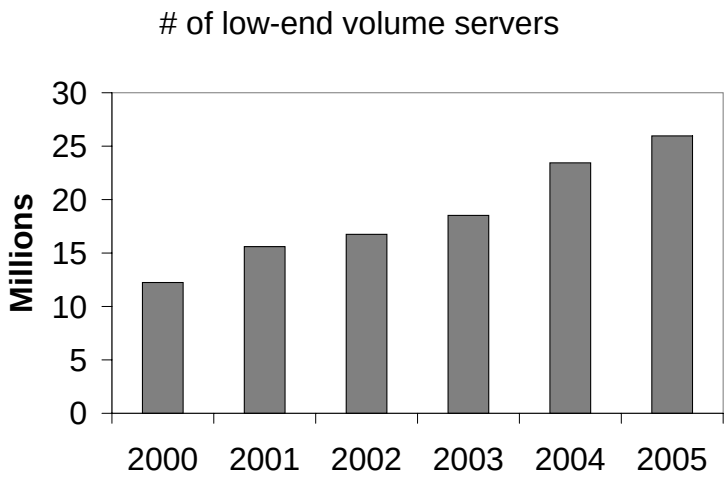
Reliable Communication for Datacenters

Mahesh Balakrishnan
Cornell University



Datacenters

- Internet Services (90s) — Websites, Search, Online Stores
- Since then:



Installed Server Base 00-05:

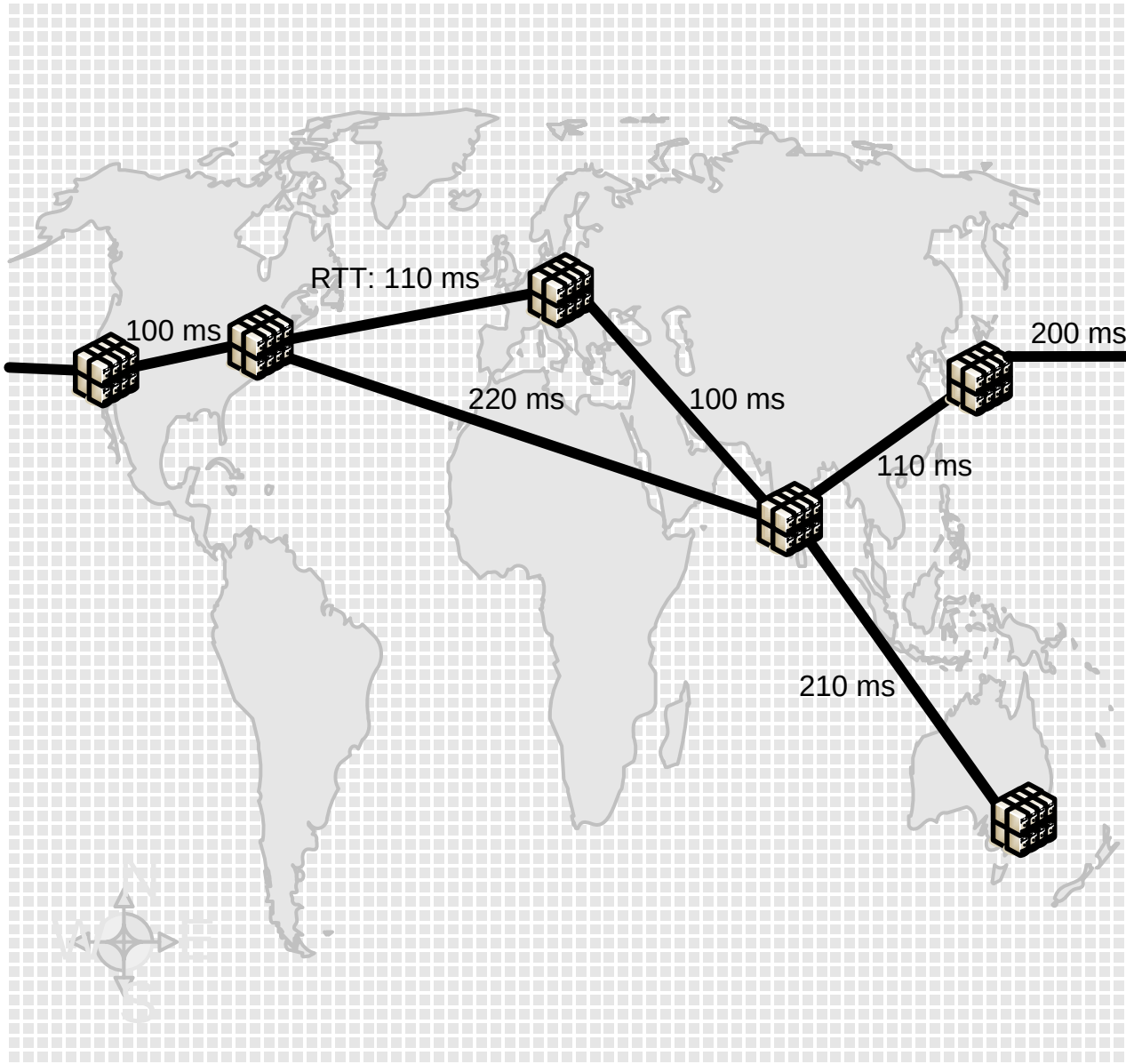
- Commodity — up by 100%
- High/Mid — down by 40%

- Today: Datacenters are ubiquitous
- How have they evolved?

Data partially sourced from IDC press releases (www.idc.com)

Networks of Datacenters

Why?
Business Continuity,
Client Locality,
Distributed Datasets
or Operations ...
Any modern
enterprise!



Networks of *Real-Time* Datacenters

- Finance, Aerospace, Military, Search and Rescue...
- ... documents, chat, email, games, videos, photos, blogs, social networks
- The Datacenter is the Computer!
- Not hard real-time: real fast, highly responsive, time-critical

Networks of *Real-Time* Datacenters

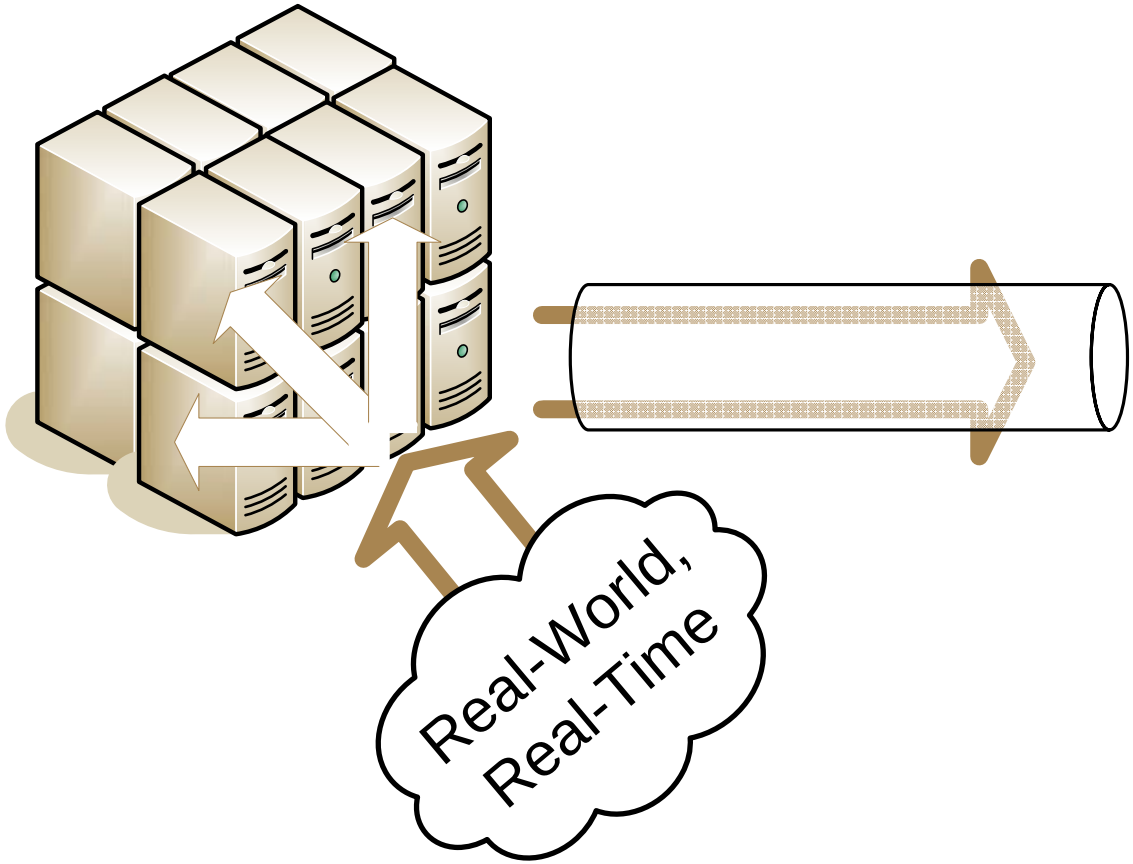
- Finance, Aerospace, Military, Search and Rescue...
- ... documents, chat, email, games, videos, photos, blogs, social networks
- The Datacenter is the Computer!
- Not hard real-time: real fast, highly responsive, time-critical

Gartner Survey:

- Real-Time Infrastructure (RTI): reaction time in secs/mins
- 73%: RTI is important or very important
- 85%: Have no RTI capability

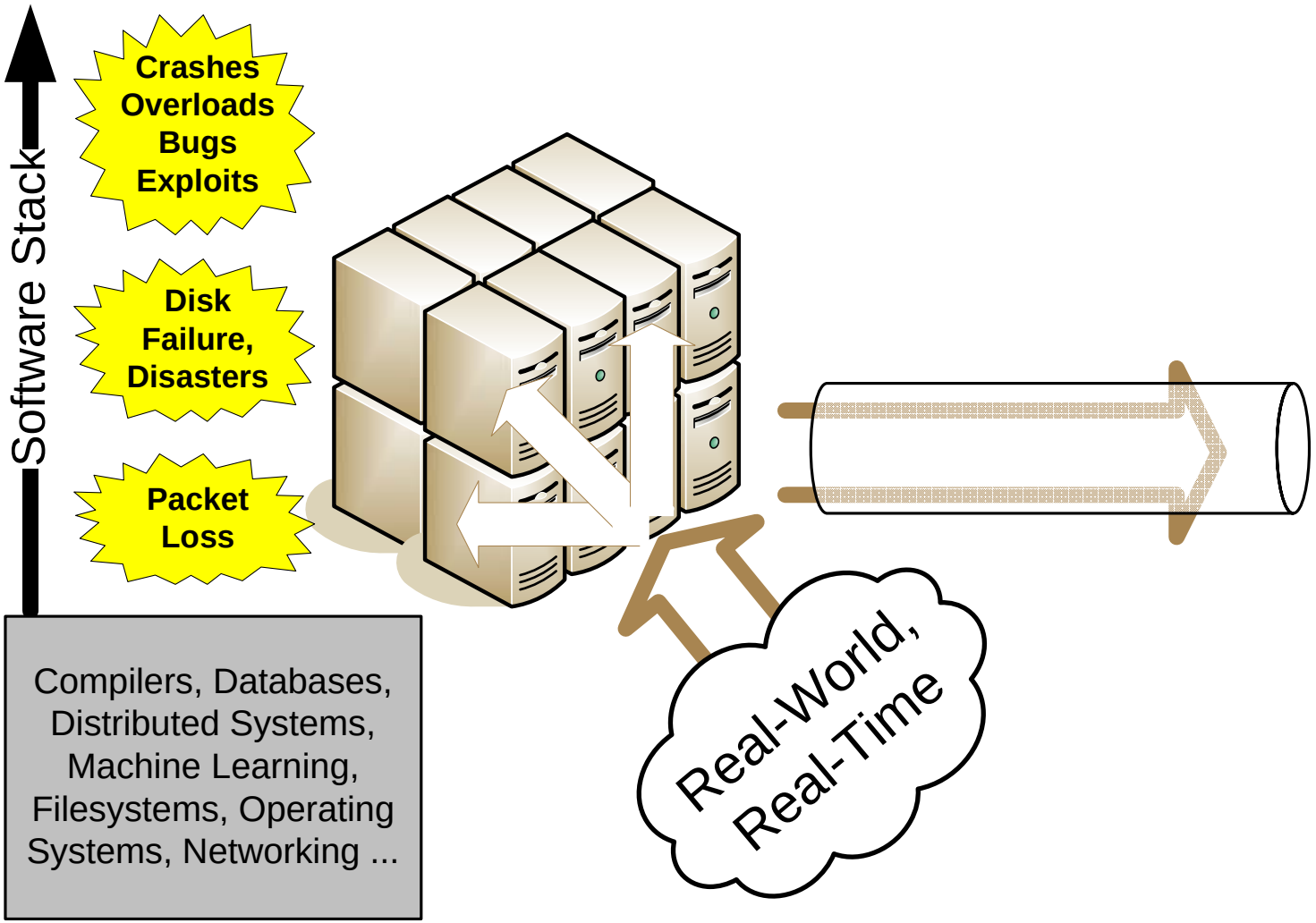
The Real-Time Datacenter — Systems Challenges

How do we recover from failures within seconds?



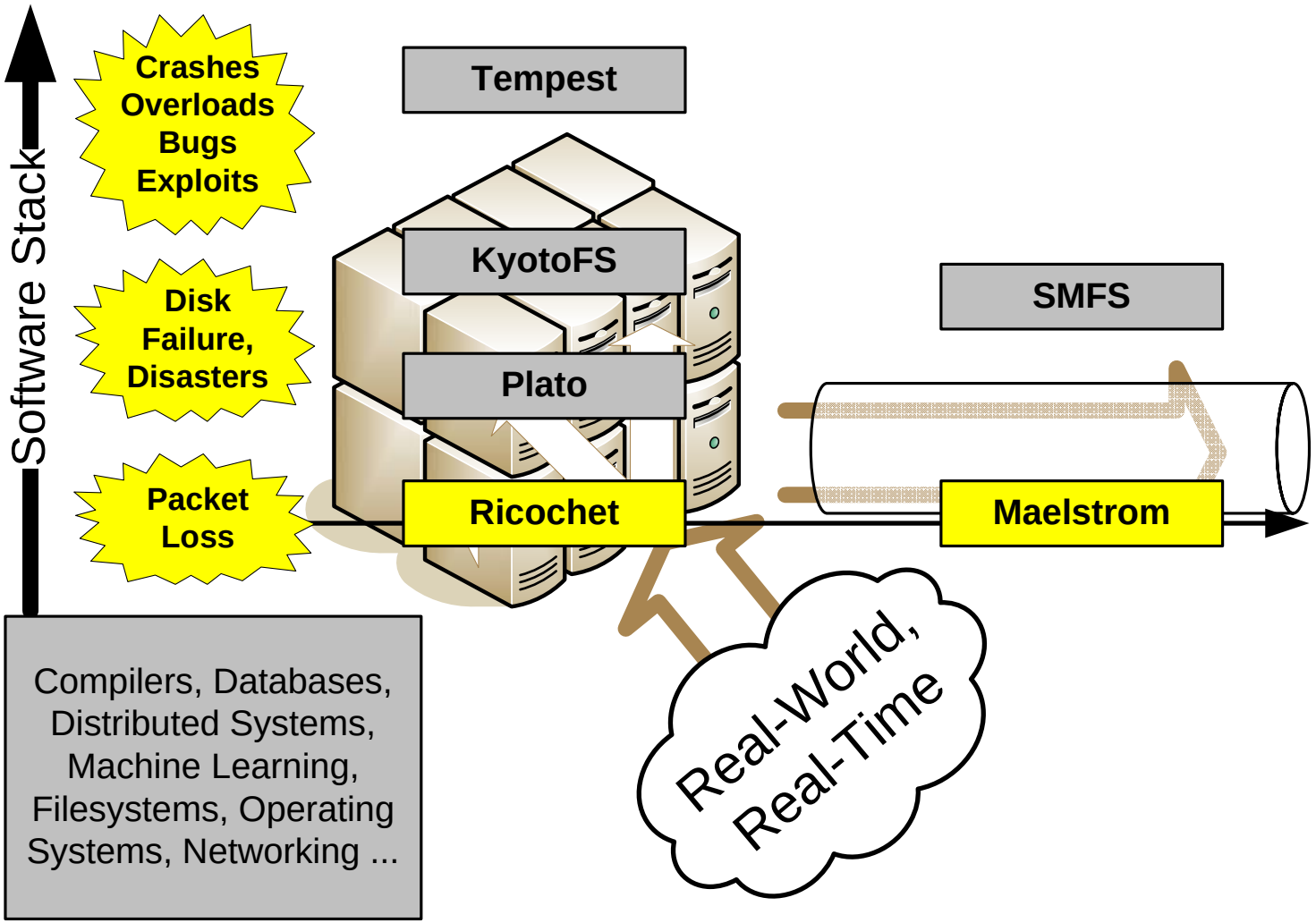
The Real-Time Datacenter — Systems Challenges

How do we recover from failures within seconds?



The Real-Time Datacenter — Systems Challenges

How do we recover from failures within seconds?



Reliable Communication

Goal: Recover lost packets *fast*!

- Existing protocols *react* to loss: too much, too late
- We want proactive recovery: stable overhead, low latencies

- Maelstrom: Reliability between datacenters [NSDI 2008]
- Ricochet: Reliability within datacenters [NSDI 2007]

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

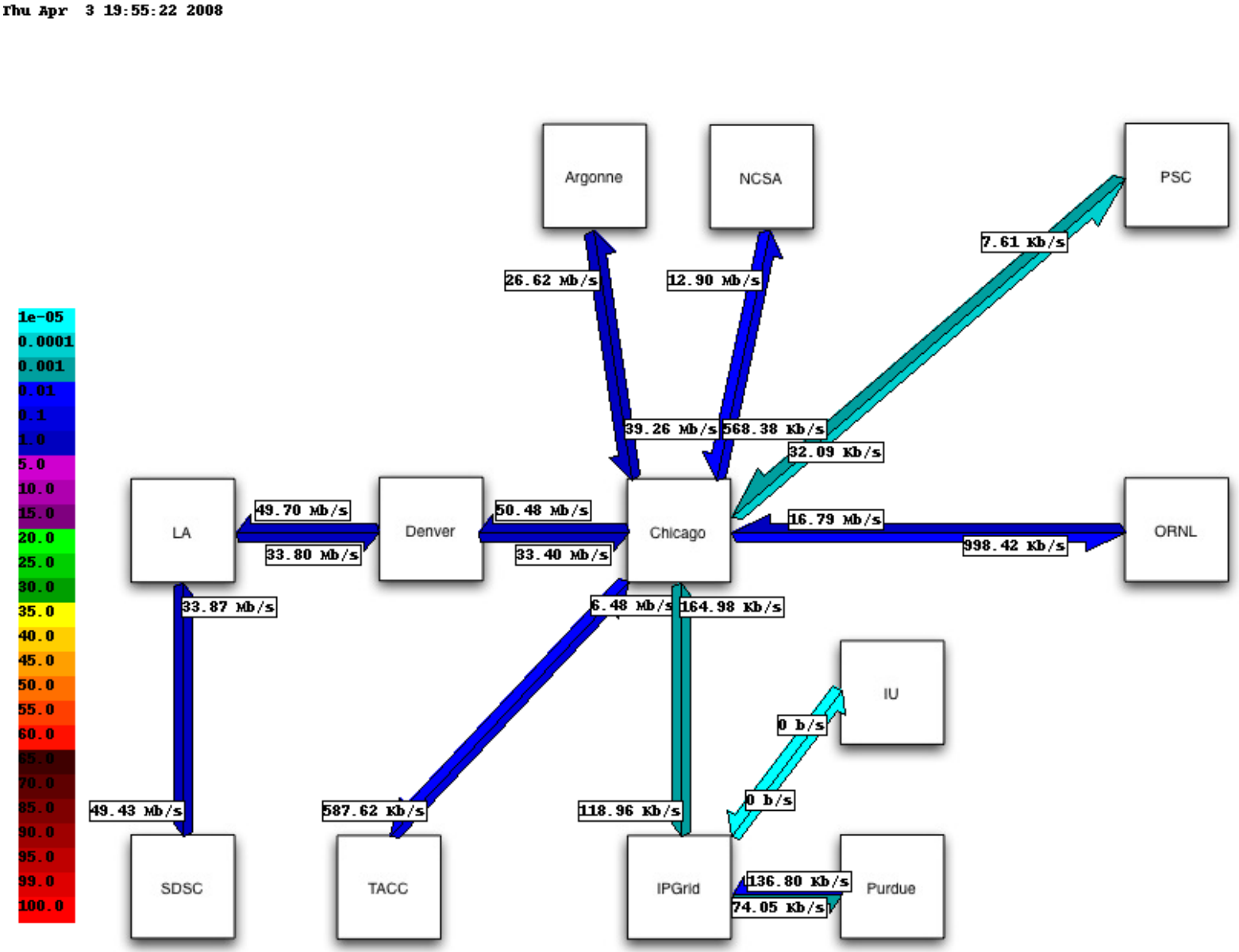
TCP fails in three ways:

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

Current Solutions:

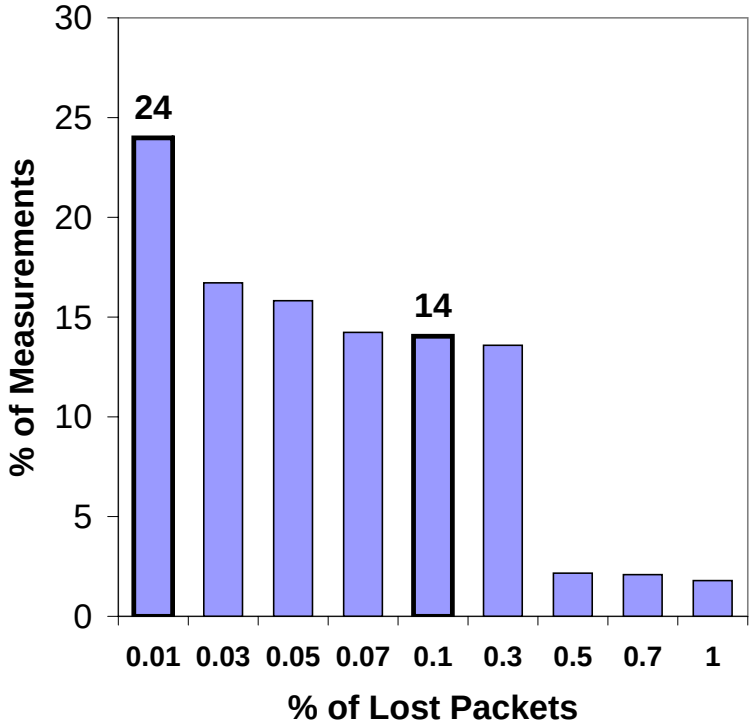
- Rewrite Apps: One Flow $\rightarrow$ Multiple Split Flows
- Resize Buffers
- *Spend (infinite) money!*

TeraGrid: Supercomputer Network



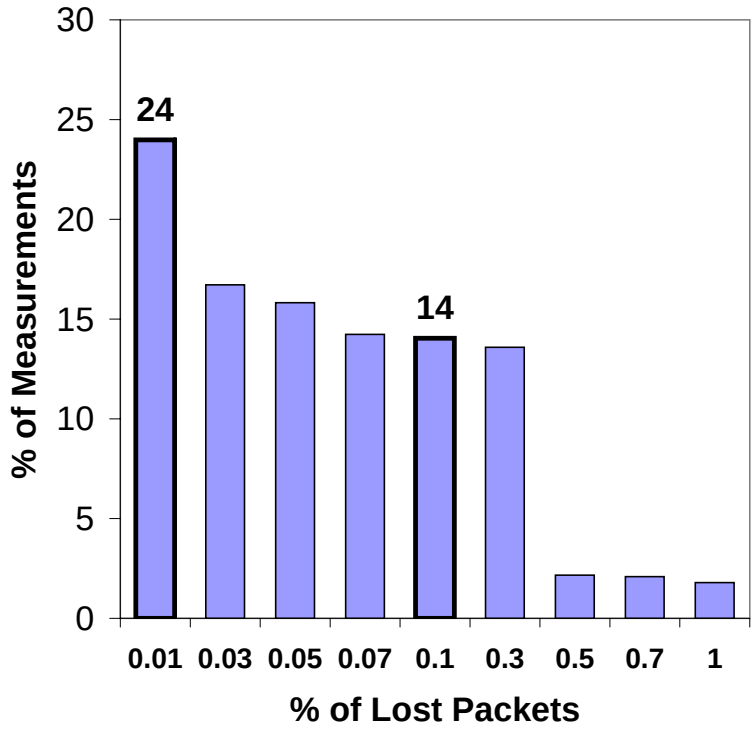
TeraGrid: Supercomputer Network

- 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
  - ...



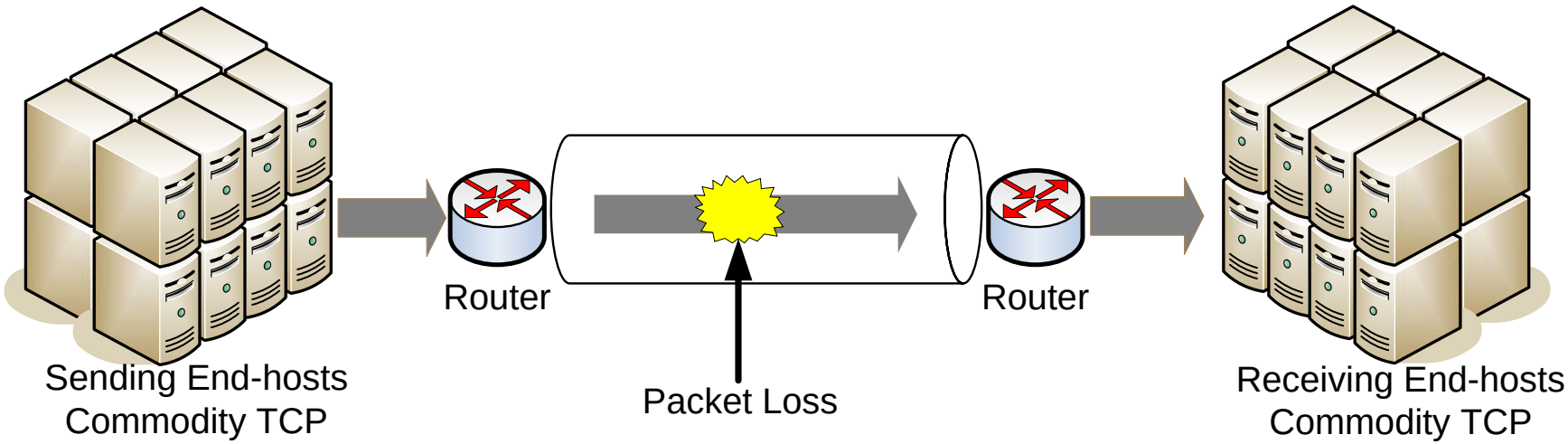
Electronics: Cluttered Pathways

Optics: Lossy Fiber

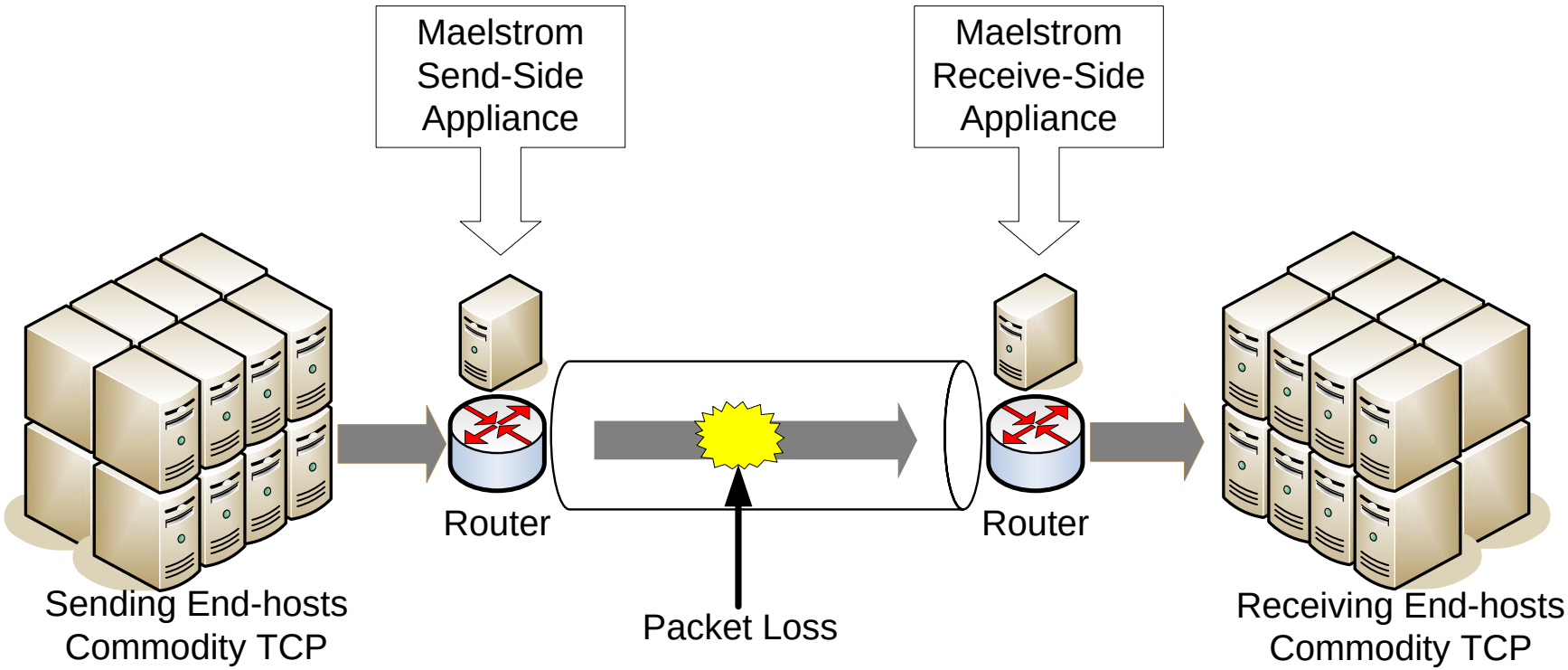
Problem Statement

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

The Maelstrom Network Appliance

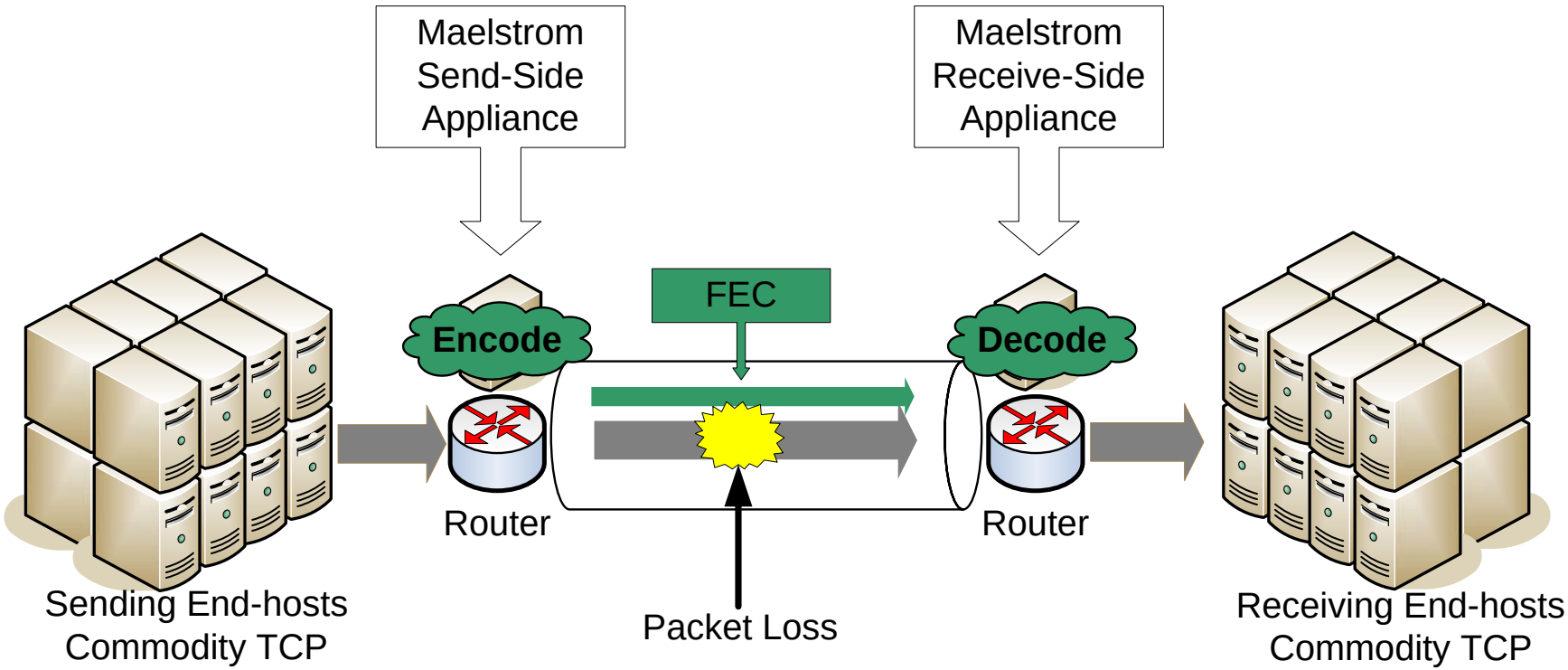


The Maelstrom Network Appliance



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 : (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

What is FEC?

3 repair packets from every 5 data packets

Receiver can recover from any 3 lost packets

Rate : (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

What is FEC?

3 repair packets from every 5 data packets

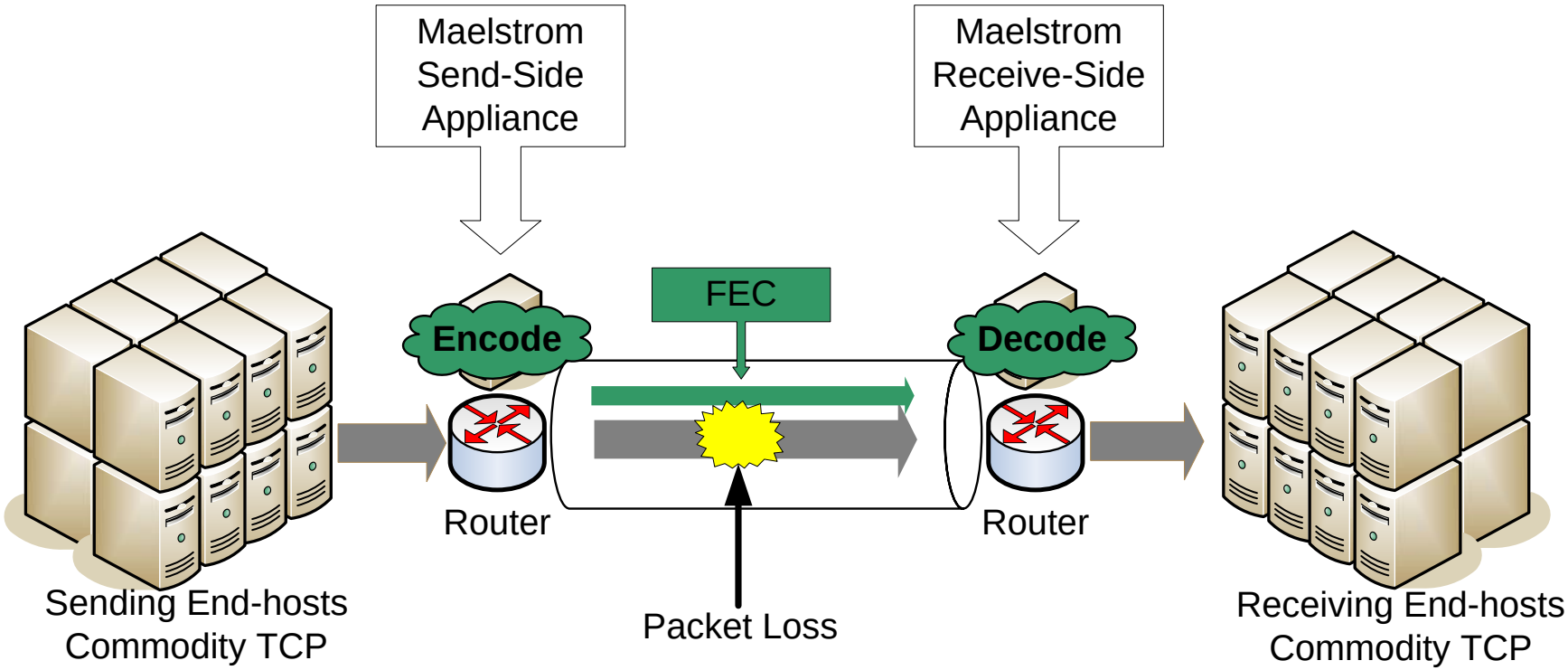
Receiver can recover from any 3 lost packets

Rate : (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

- FEC in the Network:
  - Where and What?

The Maelstrom Network Appliance



Transparent: No modification to end-host or network

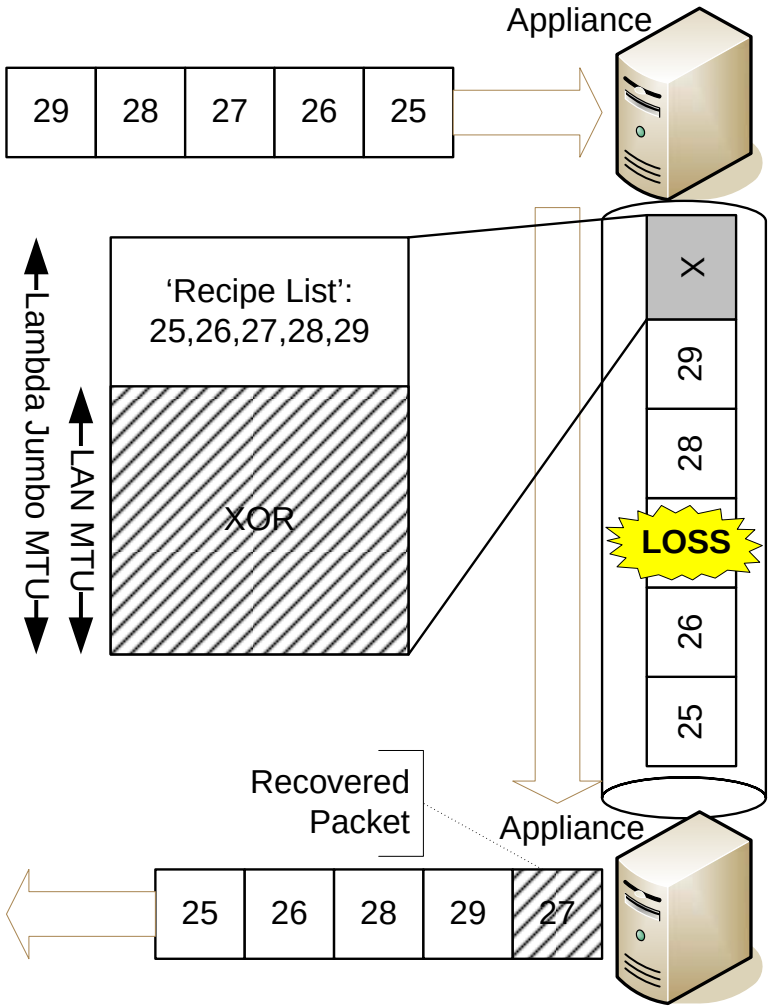
FEC = Forward Error Correction

Where: at the appliance, What: aggregated data

Maelstrom Mechanism

Send-Side Appliance:

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



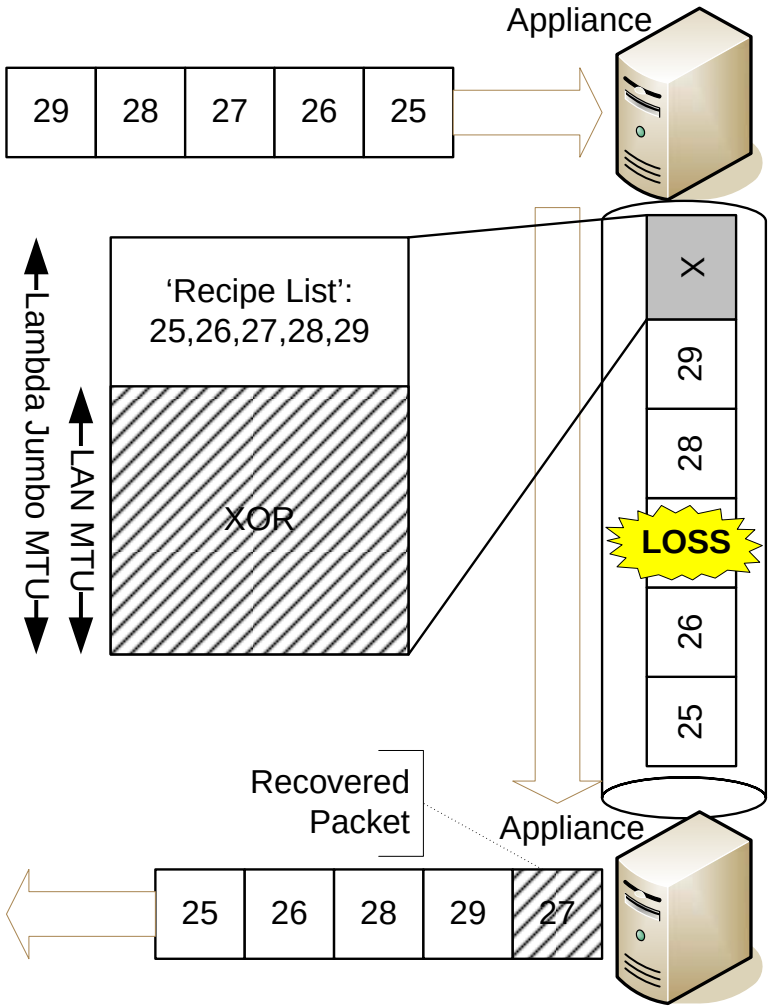
Maelstrom Mechanism

Send-Side Appliance:

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

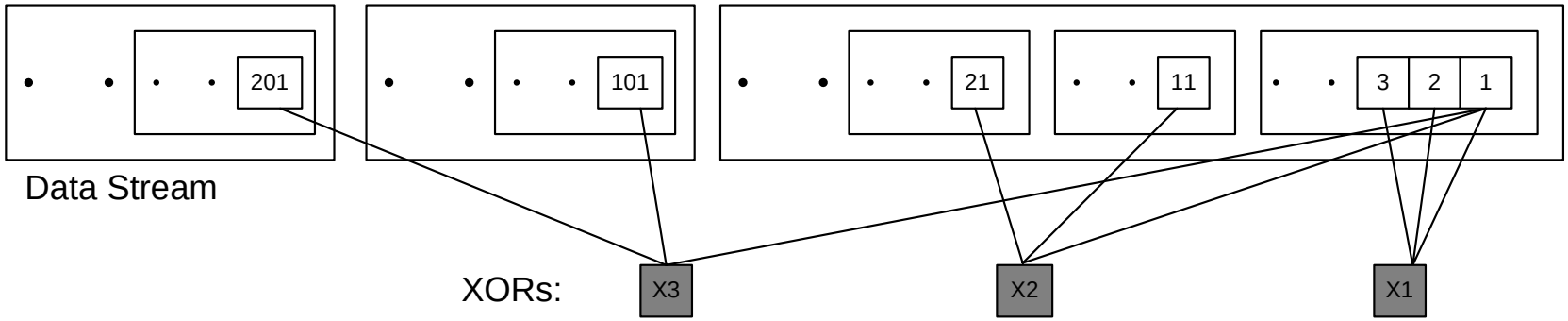
Receive-Side Appliance:

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



Layered Interleaving for Bursty Loss

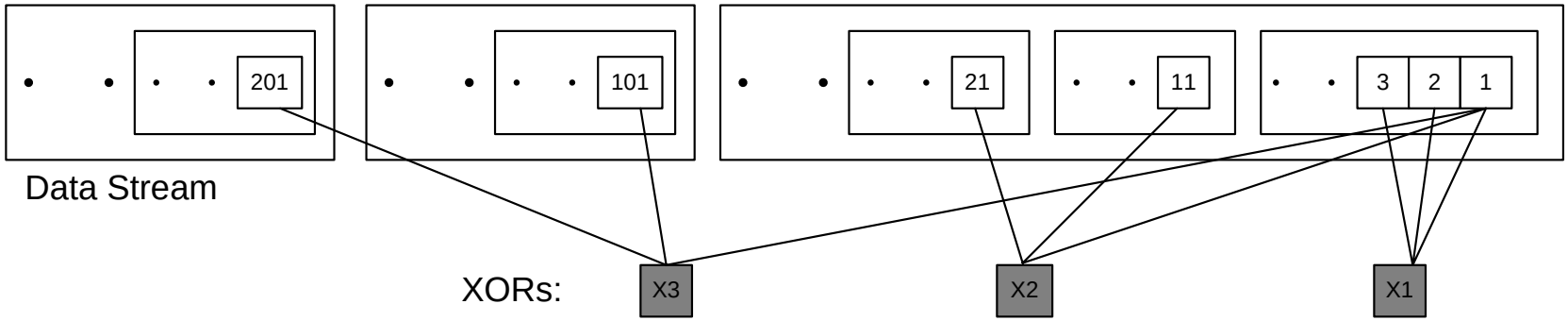
Recovery Latency $\propto$ Actual Burst Size, not Max Burst Size



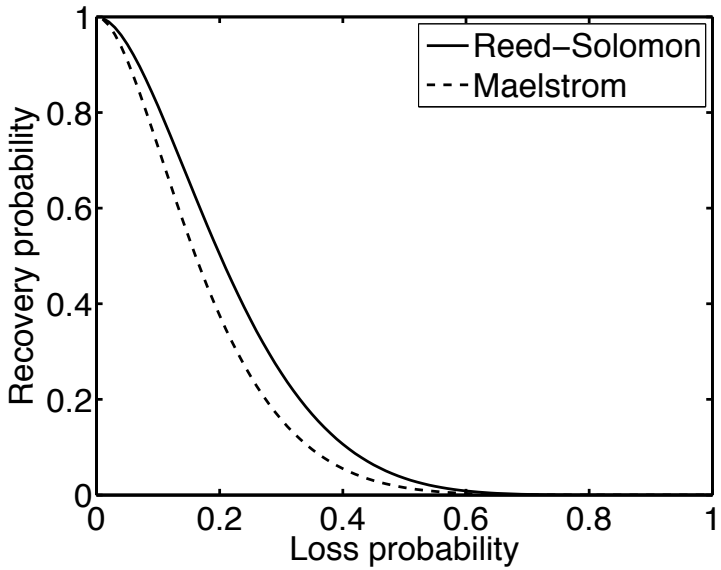
- XORs at different interleaves
- Recovery latency degrades gracefully with loss burstiness:
  X1 catches random singleton losses
  X2 catches loss bursts of 10 or less
  X3 catches bursts of 100 or less

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

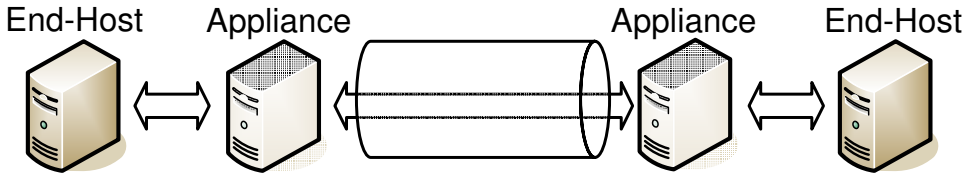


Comparison of Recovery Probability: r=7, c=2

Maelstrom Modes

- TCP Traffic: Two Flow Control Modes

A) End-to-End Flow Control



B) Split Flow Control

- Split Mode avoids client buffer resizing (PeP)

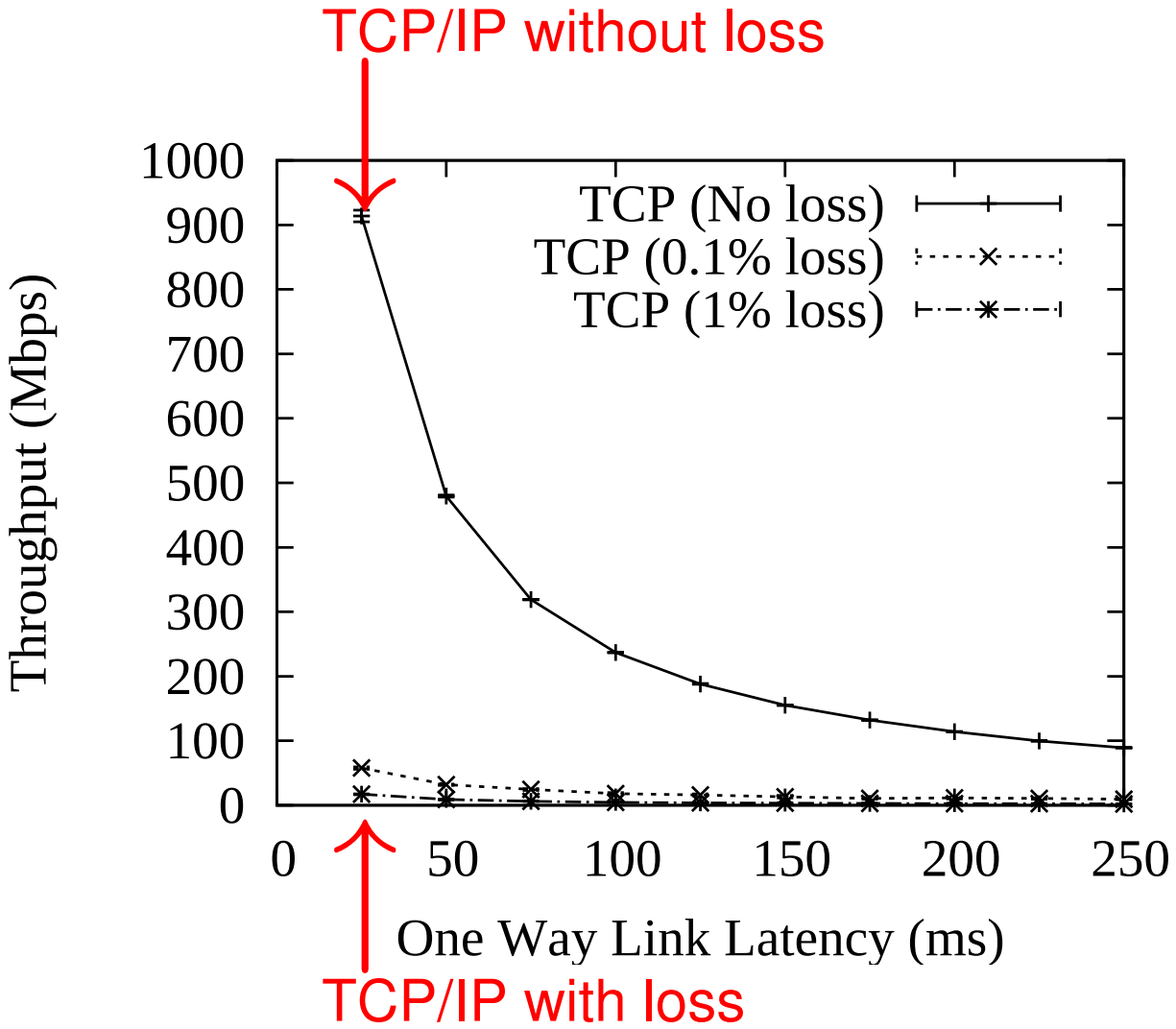
Implementation Details

- 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

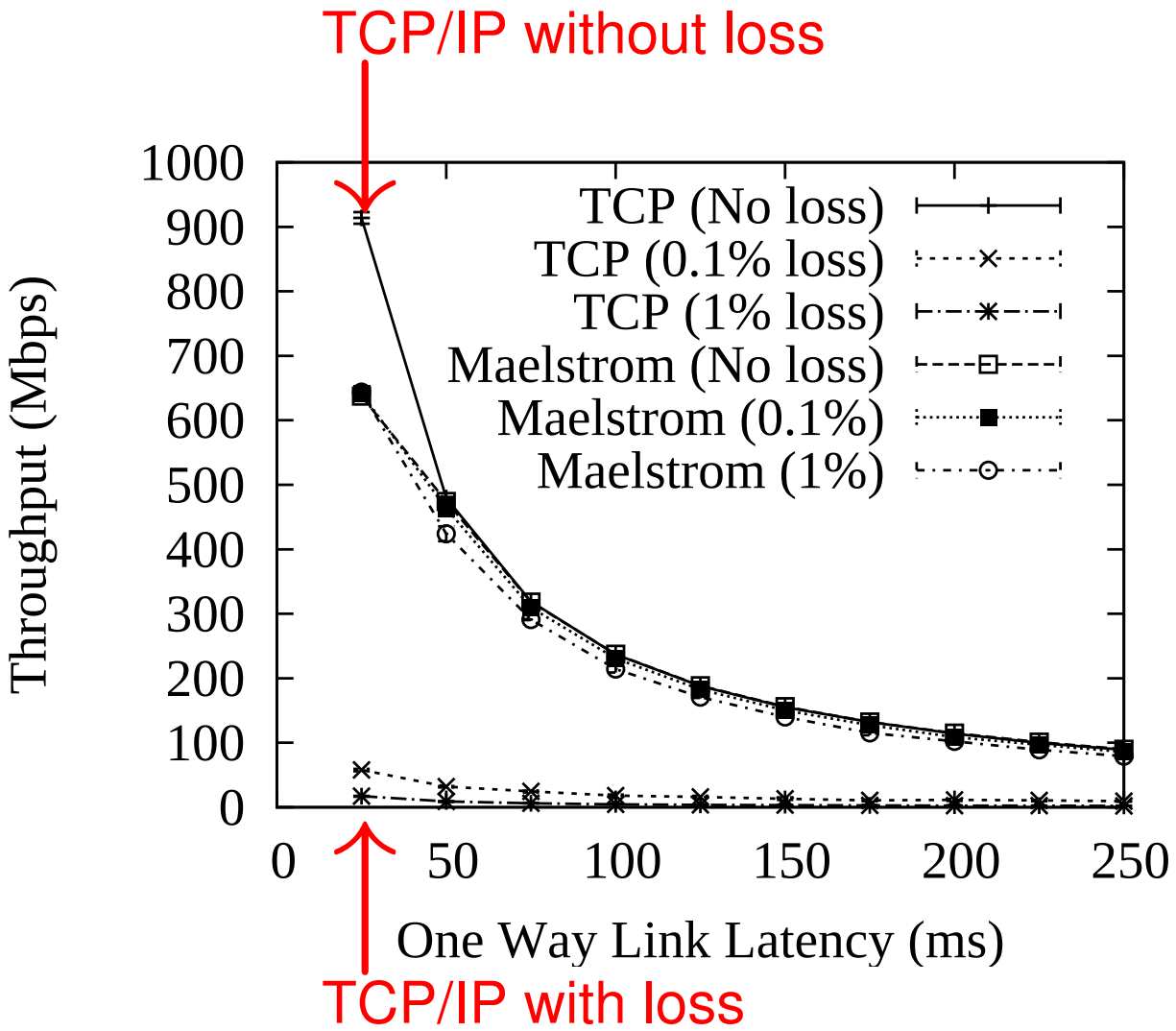
Evaluation: FEC Mode and Loss

Claim: Maelstrom effectively hides loss from TCP/IP



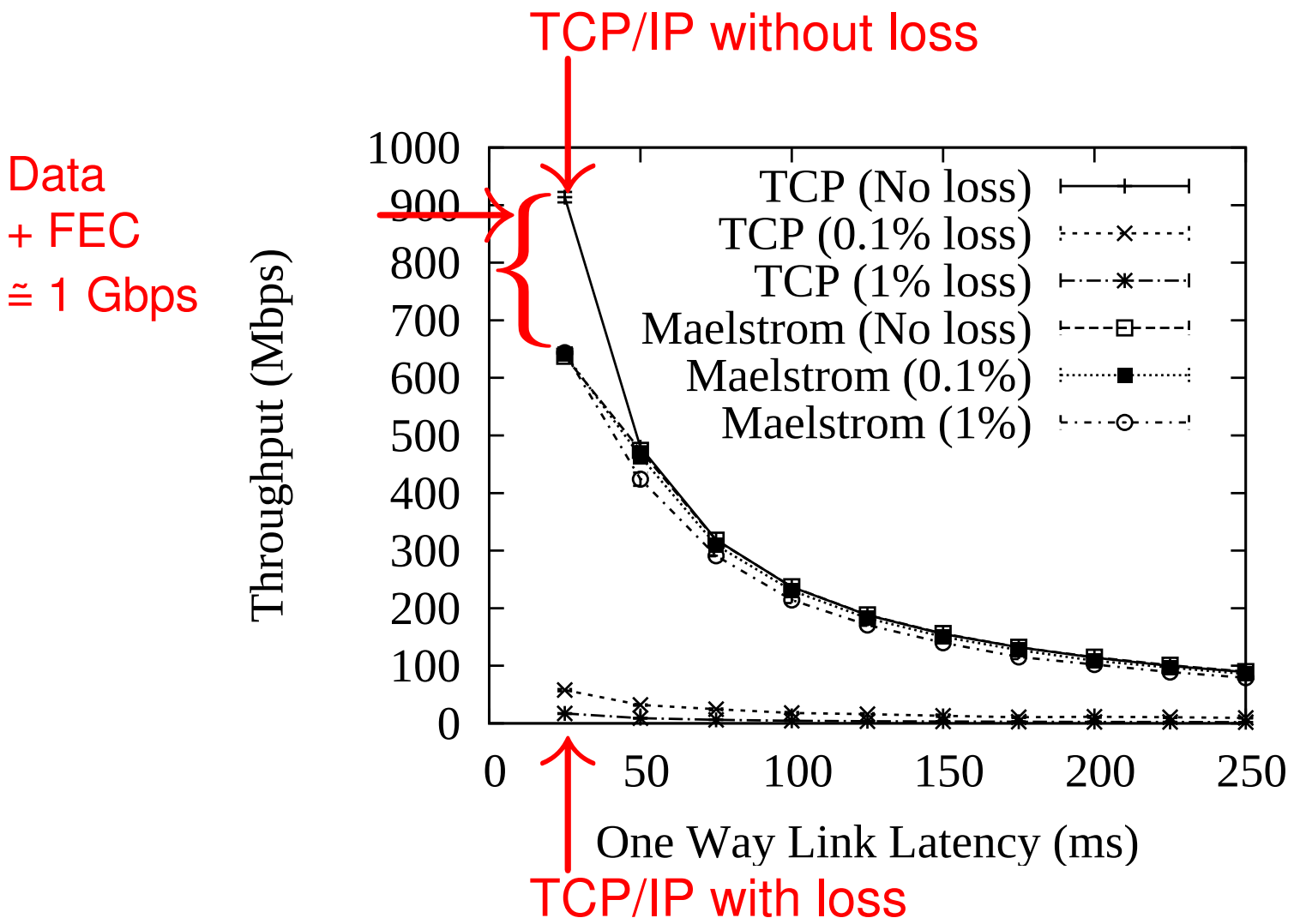
Evaluation: FEC Mode and Loss

Claim: Maelstrom effectively hides loss from TCP/IP



Evaluation: FEC Mode and Loss

Claim: Maelstrom effectively hides loss from TCP/IP

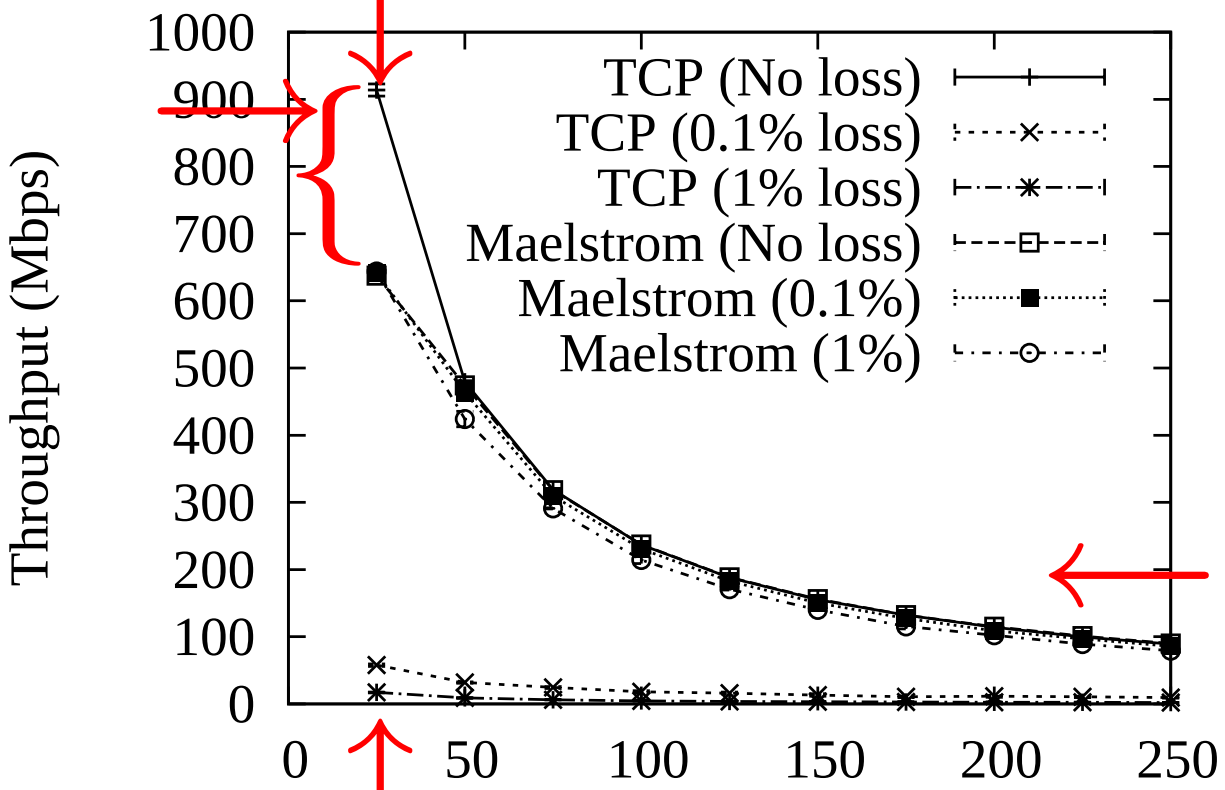


Evaluation: FEC Mode and Loss

Claim: Maelstrom effectively hides loss from TCP/IP

TCP/IP without loss

Data
+ FEC
≅ 1 Gbps

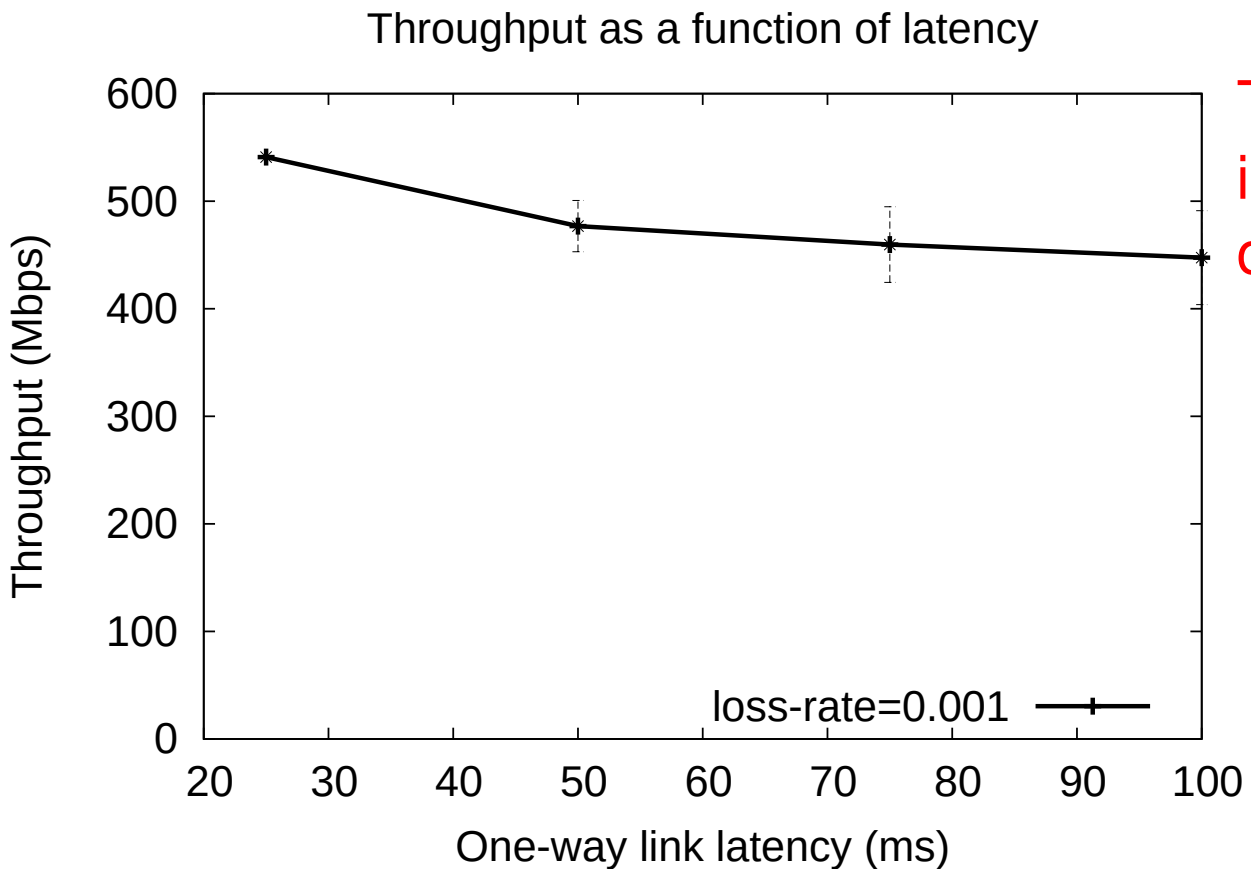


Tput $\propto \frac{1}{RTT}$

TCP/IP with loss

Evaluation: Split Mode and Buffering

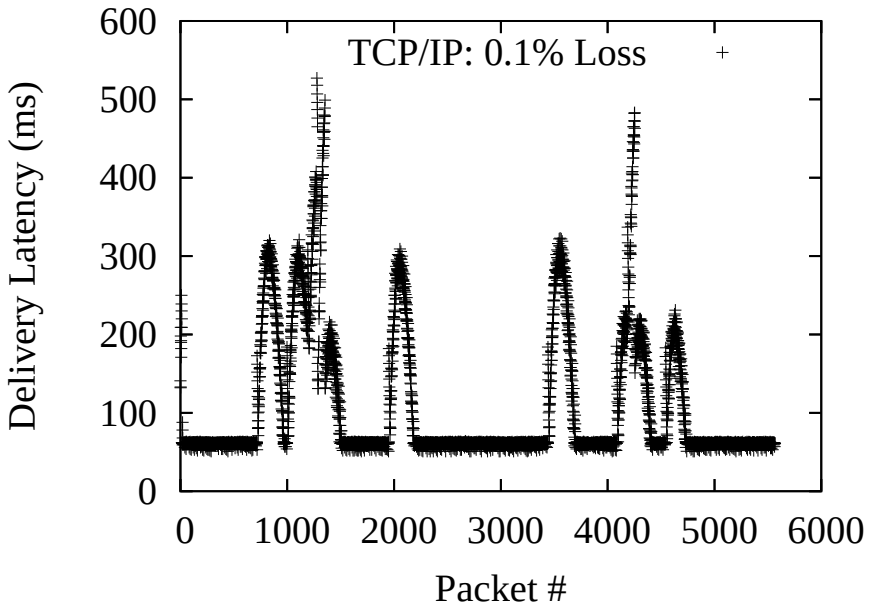
Claim: Maelstrom eliminates the need for large end-host buffers

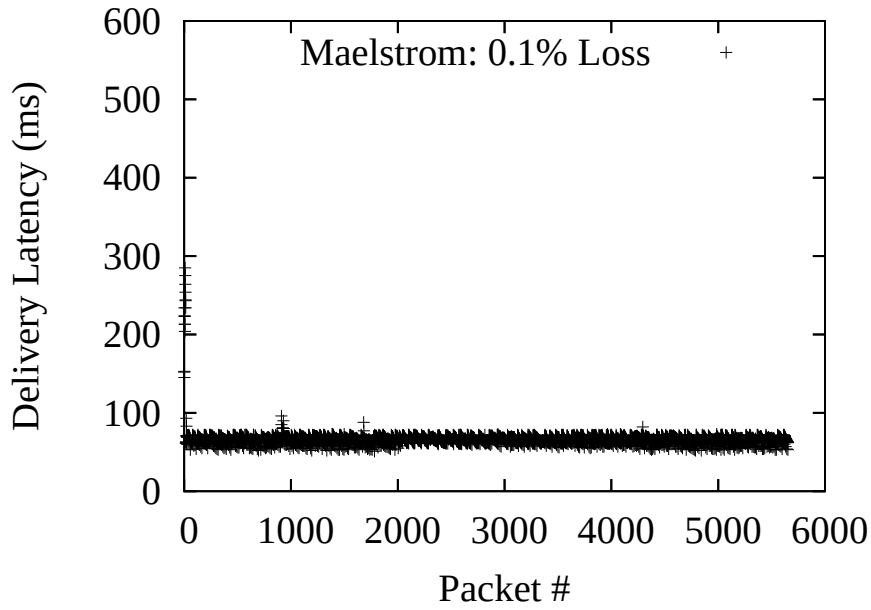


Tput independent of RTT!

Evaluation: Delivery Latency

Claim: Maelstrom eliminates TCP/IP's loss-related jitter





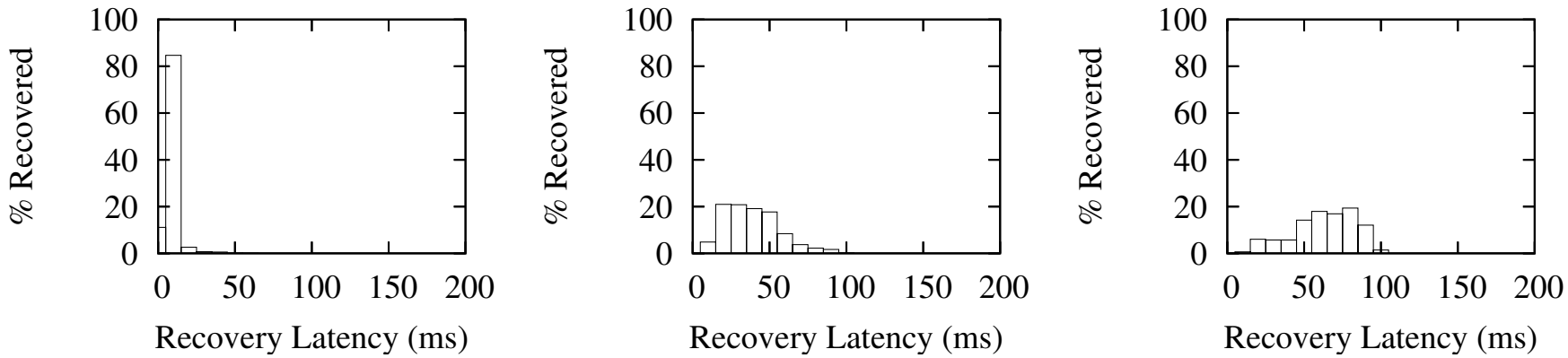
Sources of Jitter:

- 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

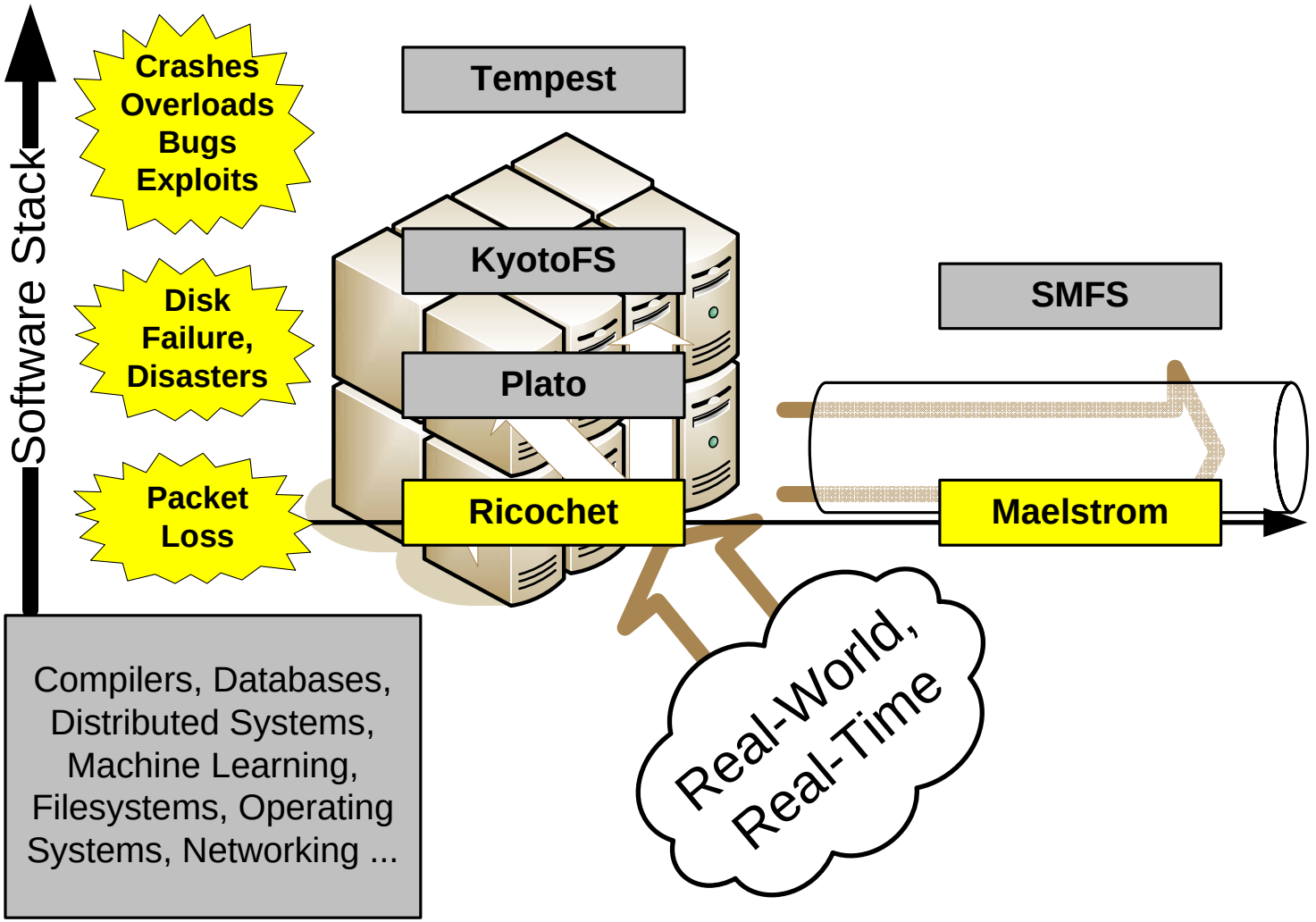


- Longer Burst Lengths → Longer Recovery Latency

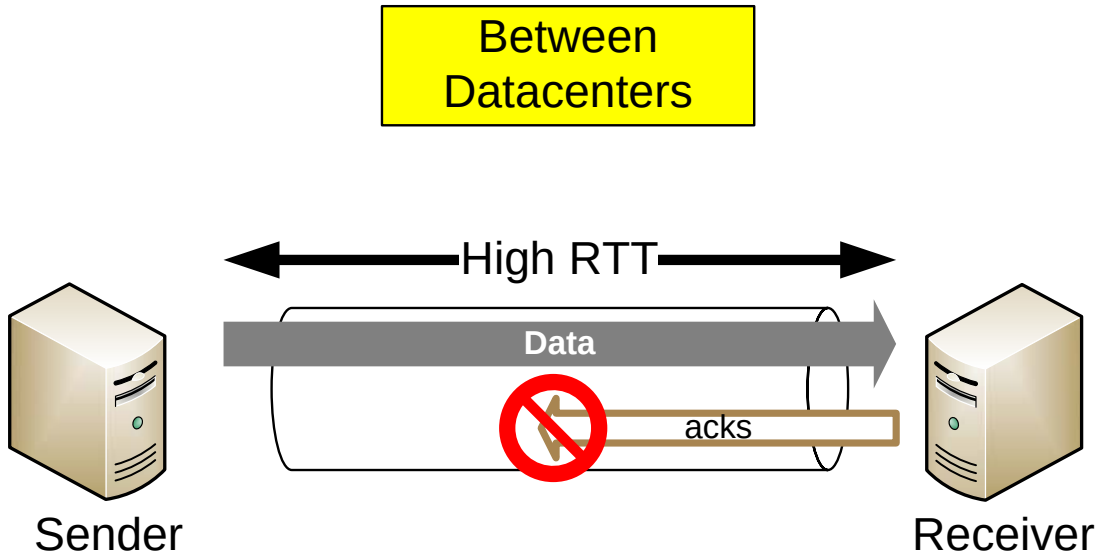
Next Step: SMFS - The Smoke and Mirrors Filesystem

- Classic Mirroring Trade-off:
  - Fast — return to user after sending to mirror
  - Safe — return to user after ACK from mirror
- SMFS — return to user after sending enough FEC
- Maelstrom: Lossy Network → Lossless Network → Disk!
- Result: Fast, Safe Mirroring independent of link length!
- General Principle: Gray-box Exposure of Protocol State

The Big Picture

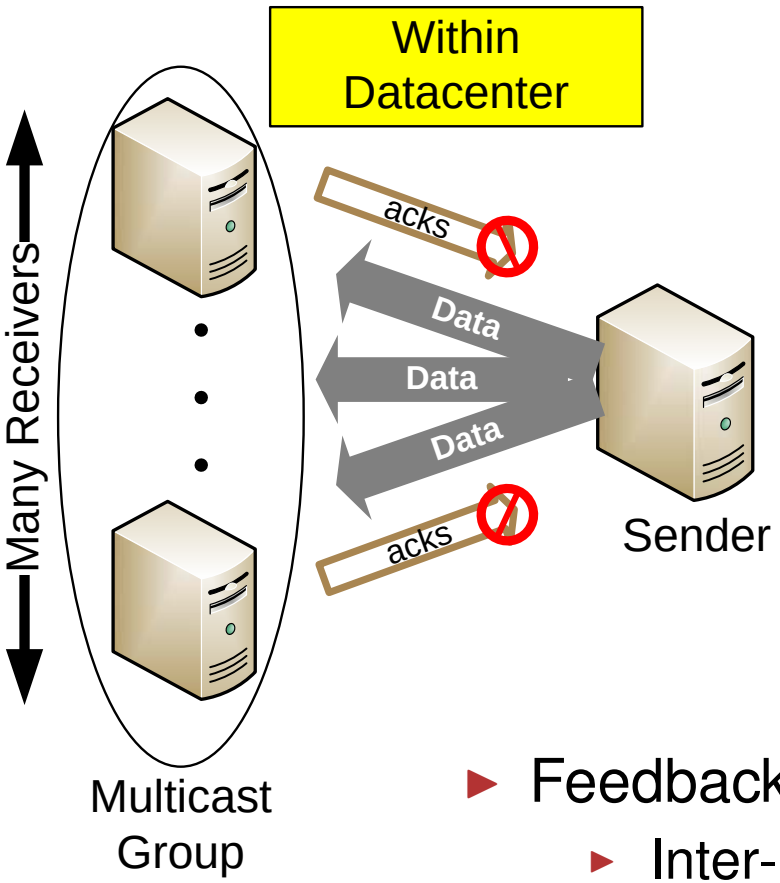


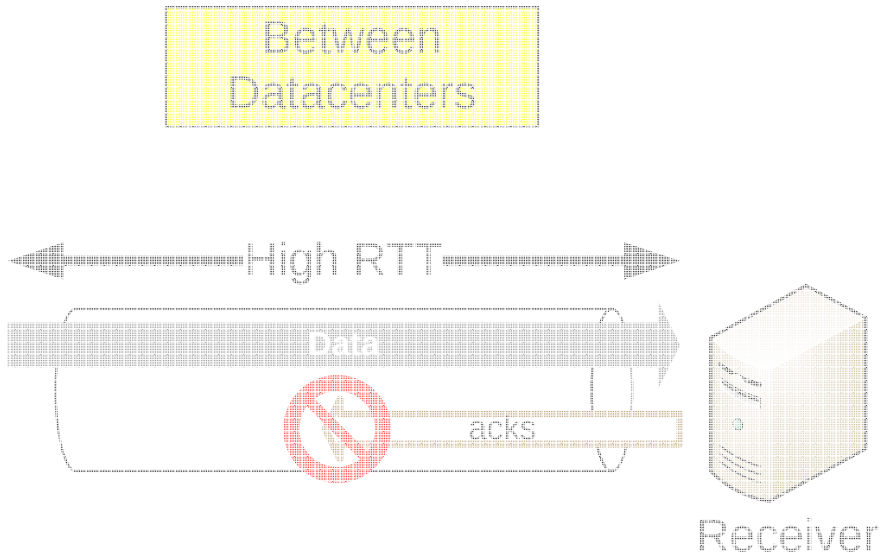
From Long-Haul to Multicast



- Feedback Loop Infeasible:
  - Inter-Datacenter Long-Haul: RTT too high

From Long-Haul to Multicast





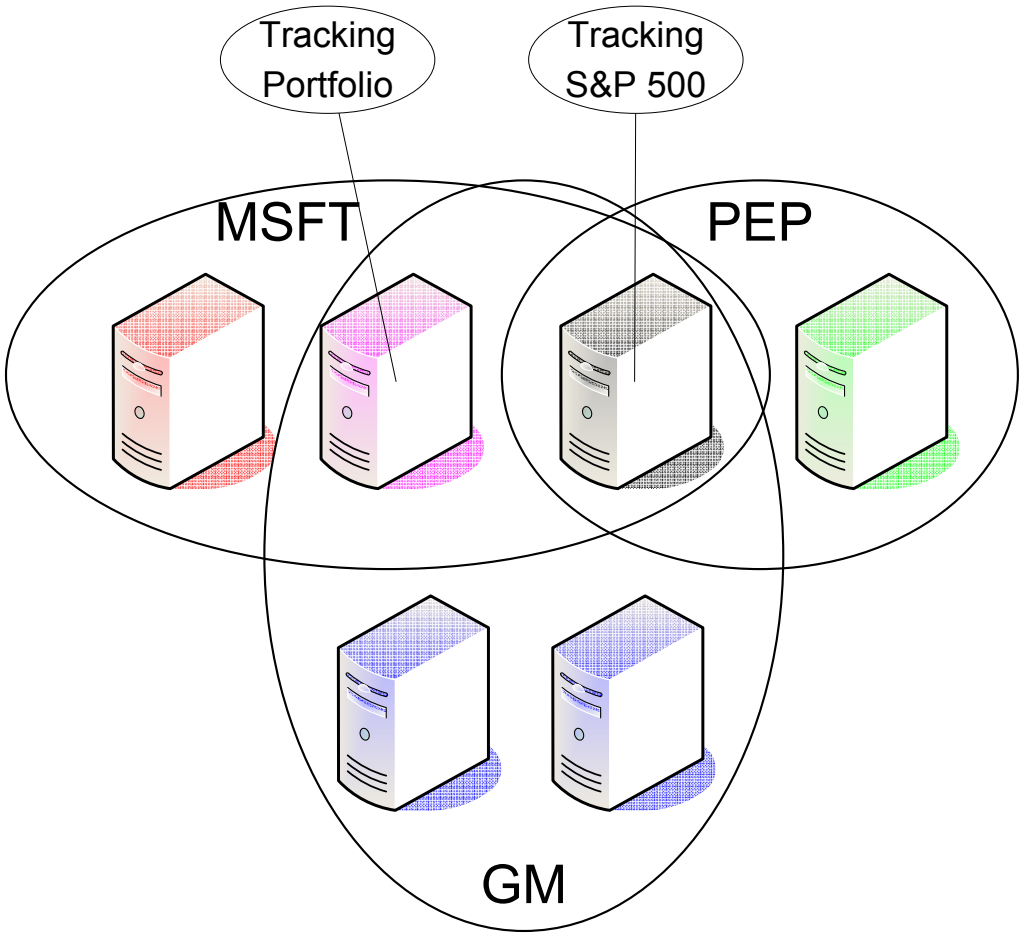
- Feedback Loop Infeasible:
  - Inter-Datacenter Long-Haul: RTT too high
  - Intra-Cluster Multicast: Too many receivers

How is Multicast Used?

service replication/partitioning, publish-subscribe, data caching...

Financial Pub-Sub Example:

- Each equity is mapped to a multicast group
- Each node is interested in a different set of equities ...



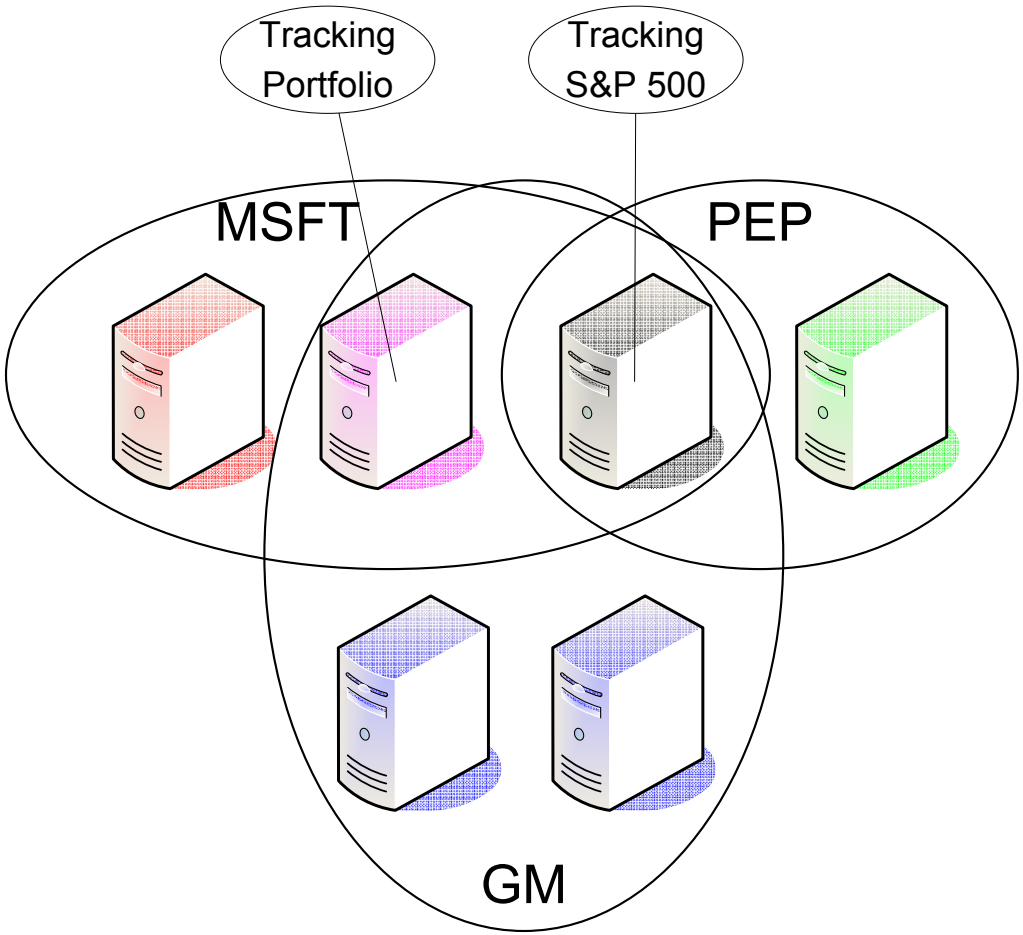
How is Multicast Used?

service replication/partitioning, publish-subscribe, data caching...

Financial Pub-Sub Example:

- Each equity is mapped to a multicast group
- Each node is interested in a different set of equities ...

Each node in many groups
$\Longrightarrow$ Low per-group data rate



How is Multicast Used?

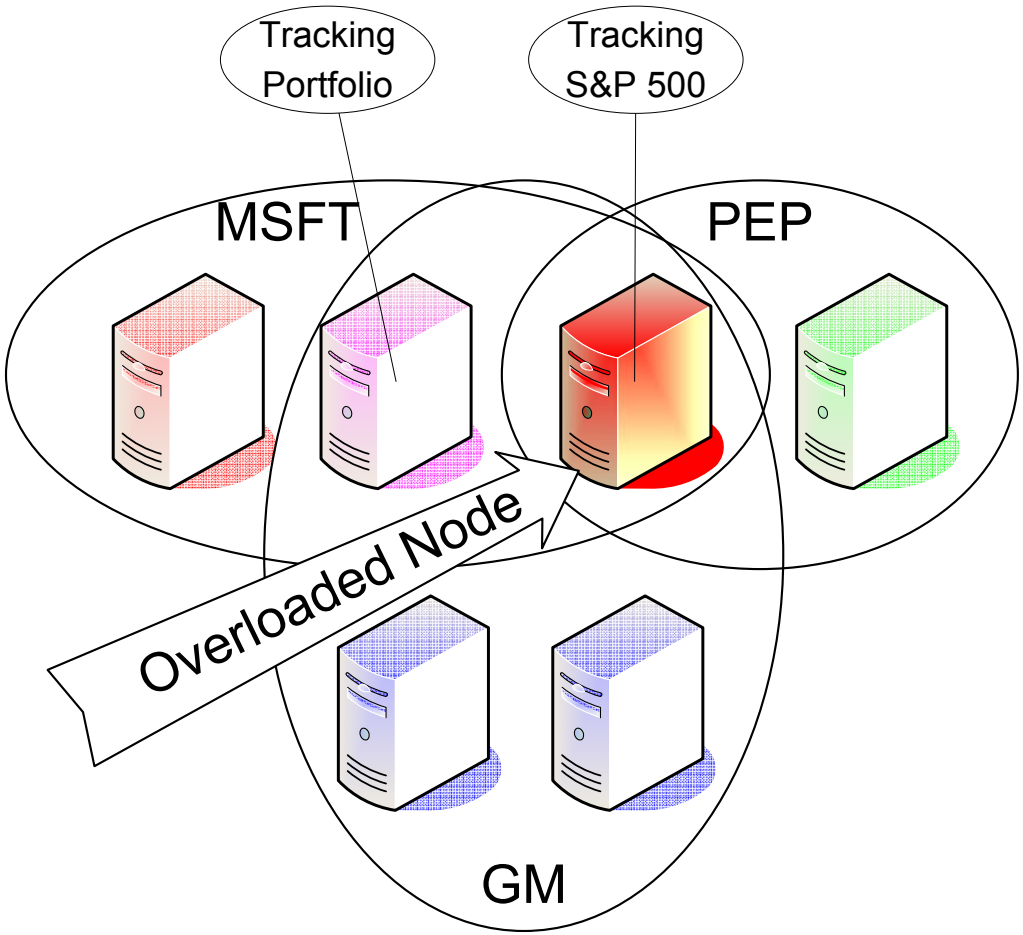
service replication/partitioning, publish-subscribe, data caching...

Financial Pub-Sub Example:

- Each equity is mapped to a multicast group
- Each node is interested in a different set of equities ...

Each node in many groups
$\Longrightarrow$ Low per-group data rate

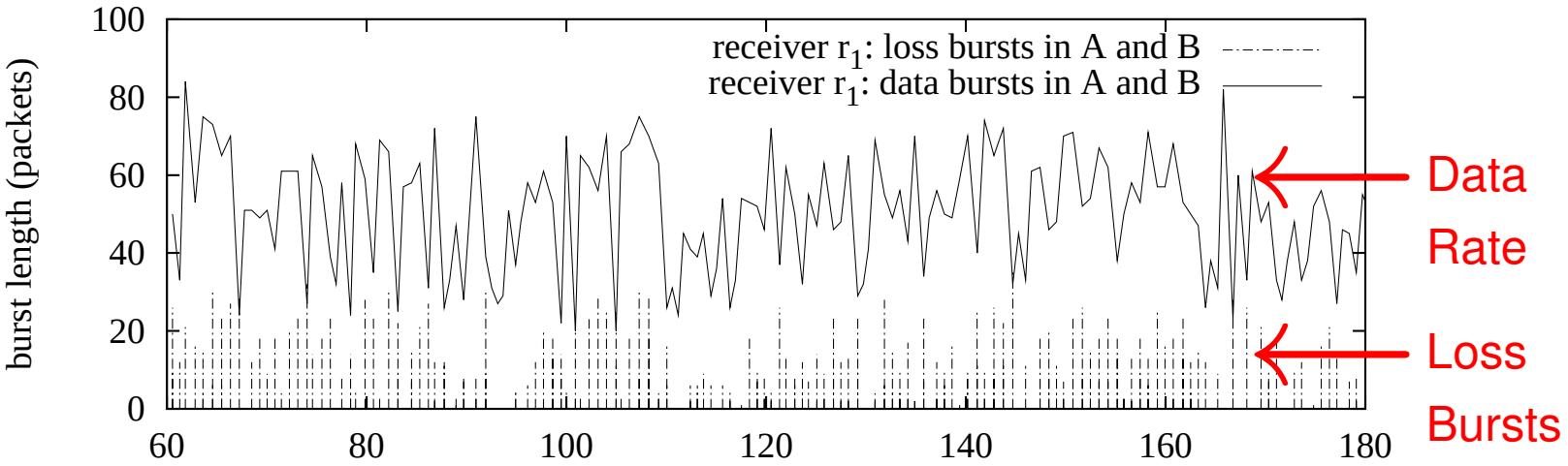
High per-node data rate
$\Longrightarrow$ Overload



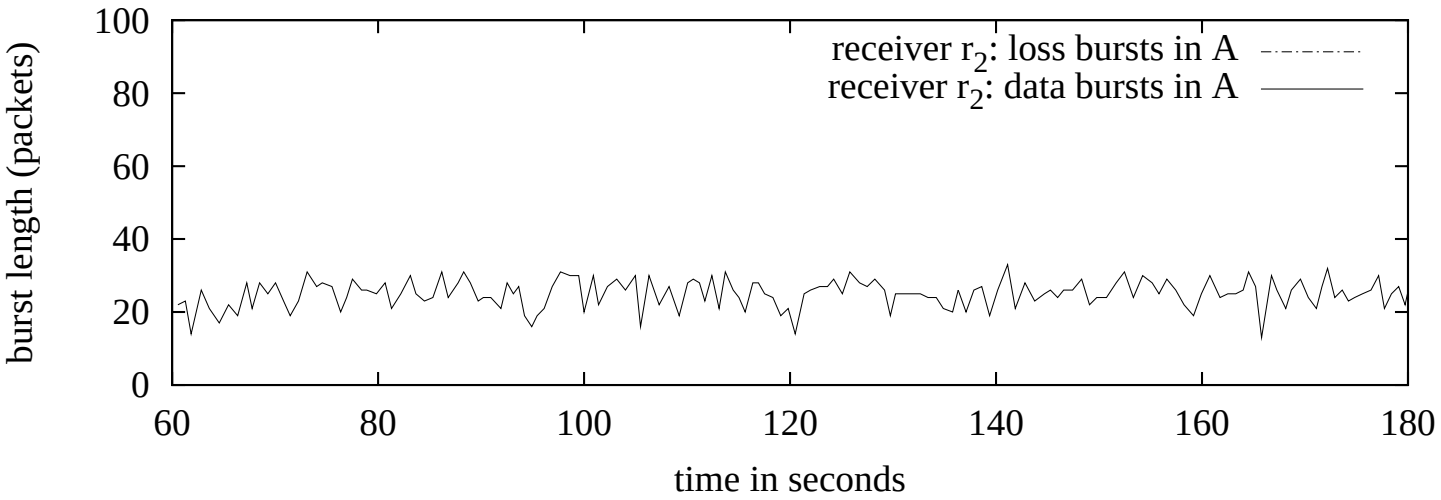
Where does loss occur in a Datacenter?

Packet Loss occurs at end-hosts: independent and bursty

Overloaded Node

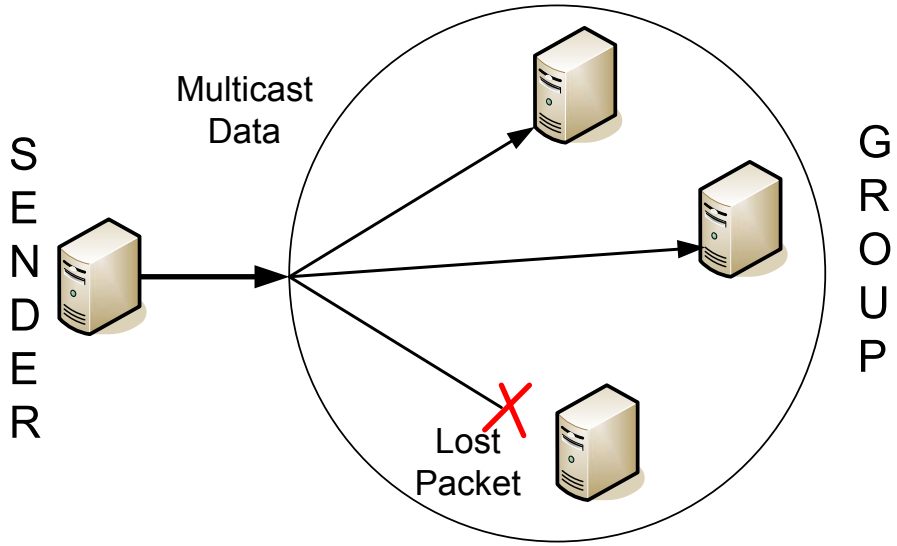


Less Loaded Node



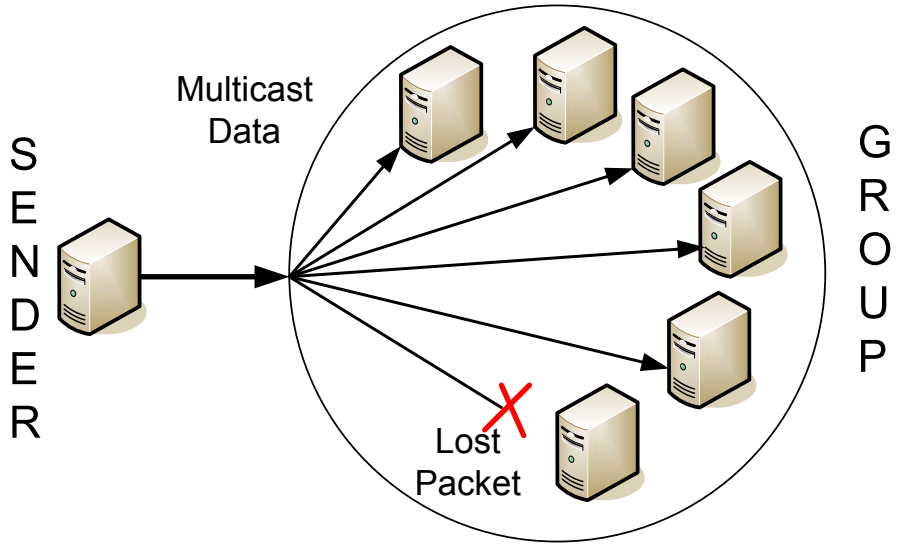
Problem Statement

- Recover lost packets rapidly!
- Scalability:



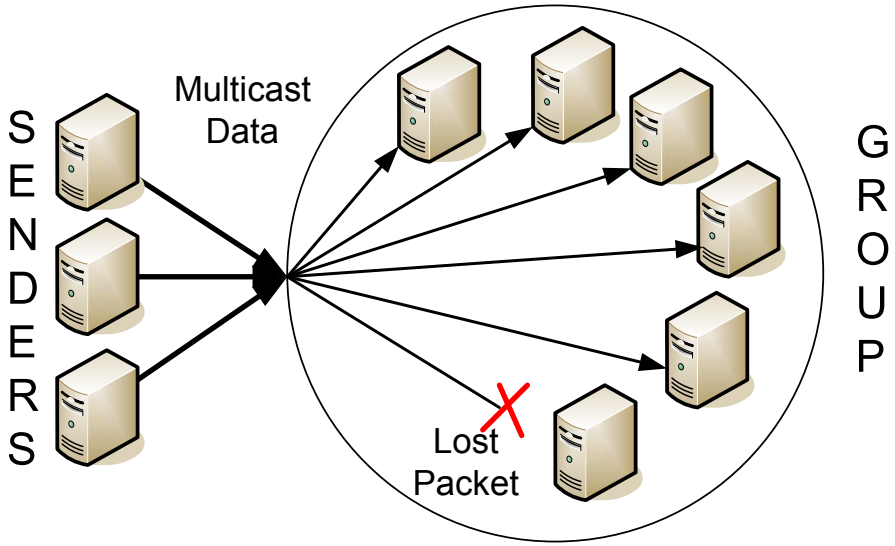
Problem Statement

- Recover lost packets rapidly!
- Scalability:
  - Number of Receivers



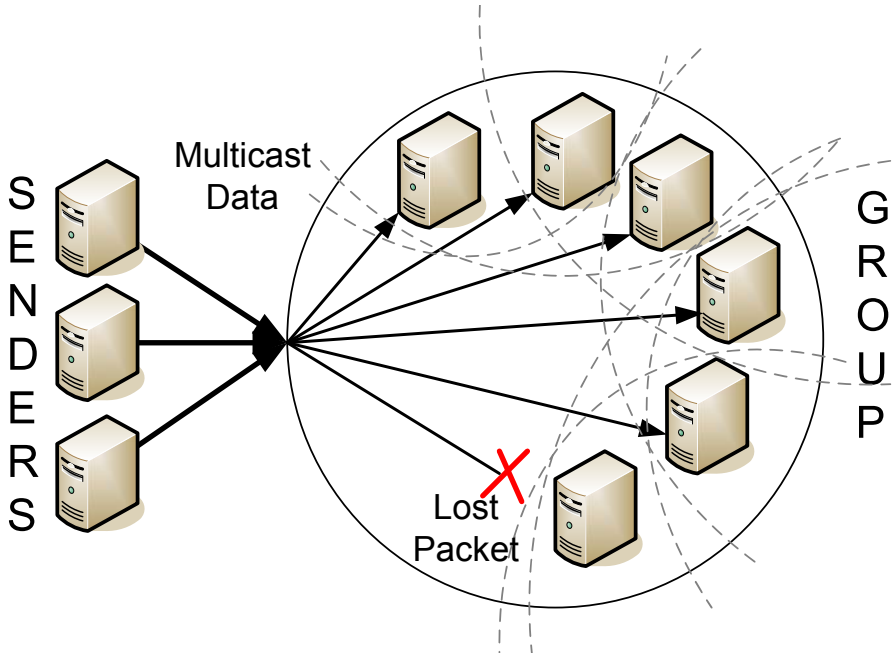
Problem Statement

- Recover lost packets rapidly!
- Scalability:
  - Number of Receivers
  - Number of Senders



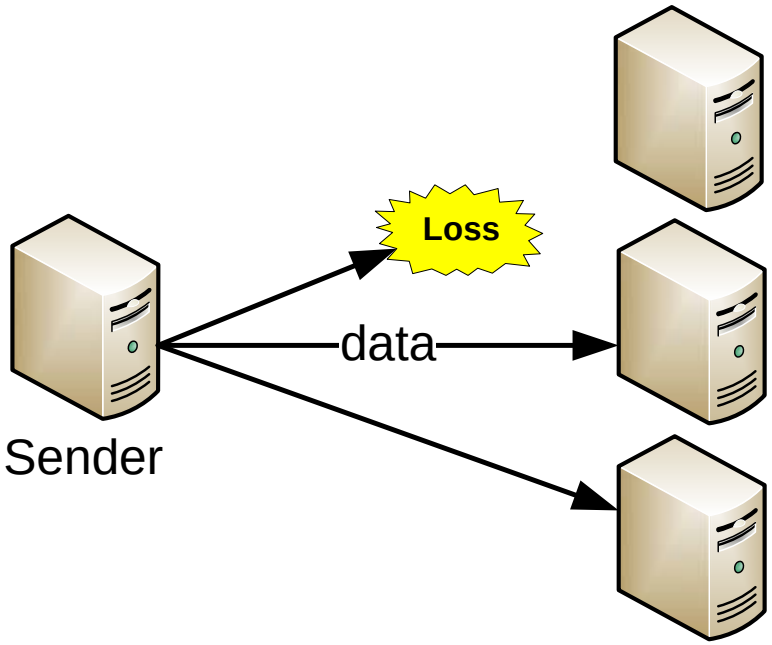
Problem Statement

- Recover lost packets rapidly!
- Scalability:
  - Number of Receivers
  - Number of Senders
  - Number of Groups



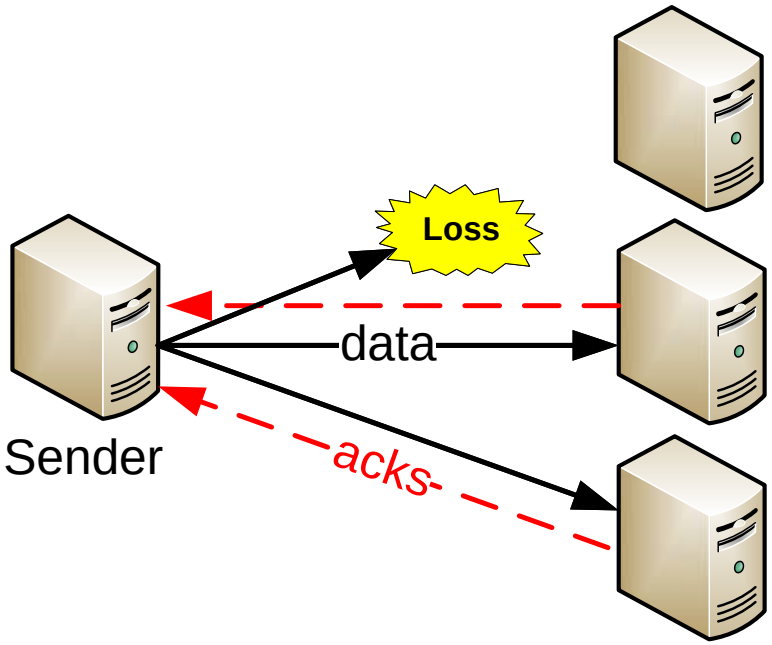
Design Space for Reliable Multicast

How does latency scale?



Design Space for Reliable Multicast

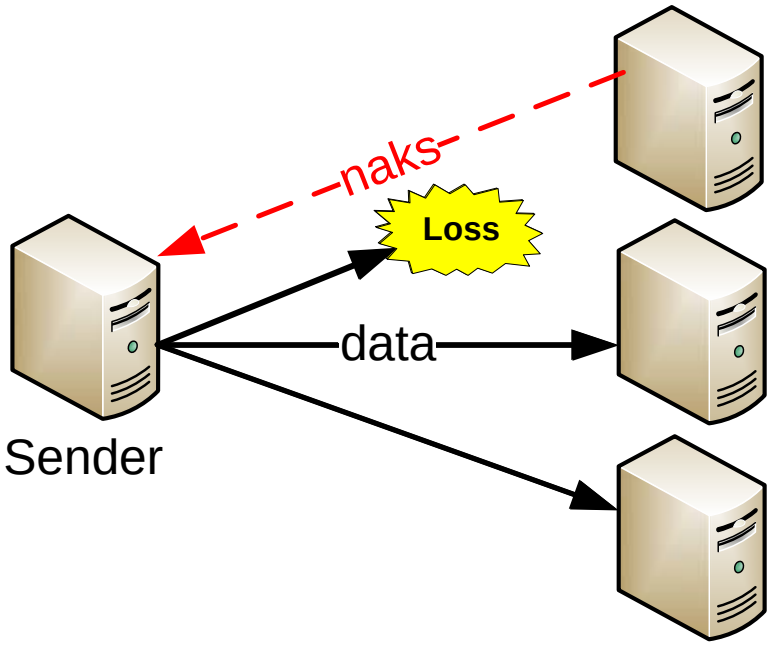
How does latency scale?



- 1. acks: implosion

Design Space for Reliable Multicast

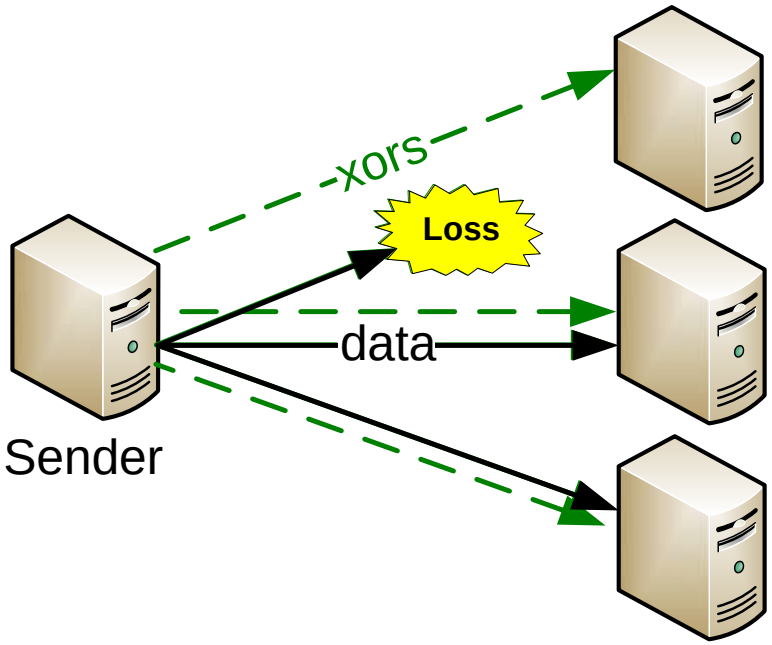
How does latency scale?



- 1. acks: implosion
- 2. naks

Design Space for Reliable Multicast

How does latency scale?



- 1. acks: implosion
- 2. naks
- 3. Sender-based FEC

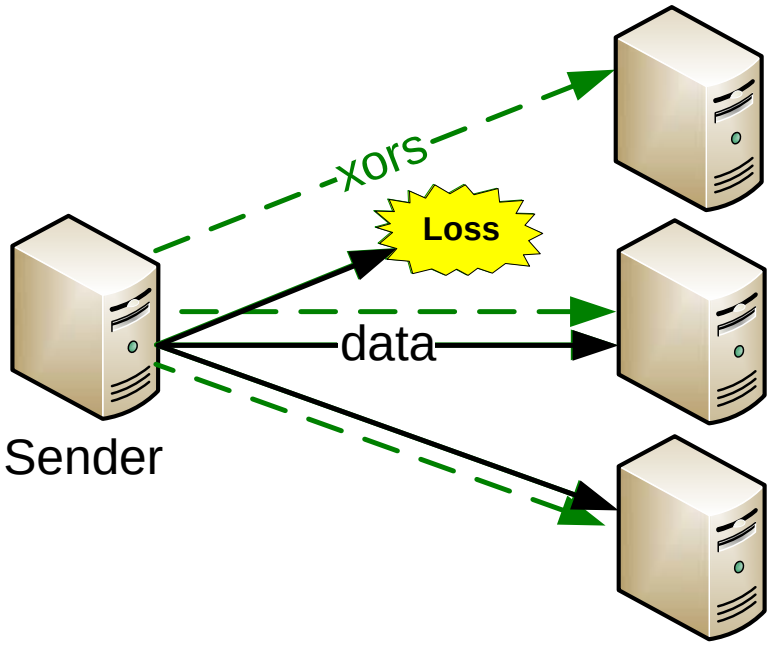
  $\textit{recovery latency} \propto \frac{1}{\textit{datarate}}$

  data rate:

  one sender, one group

Design Space for Reliable Multicast

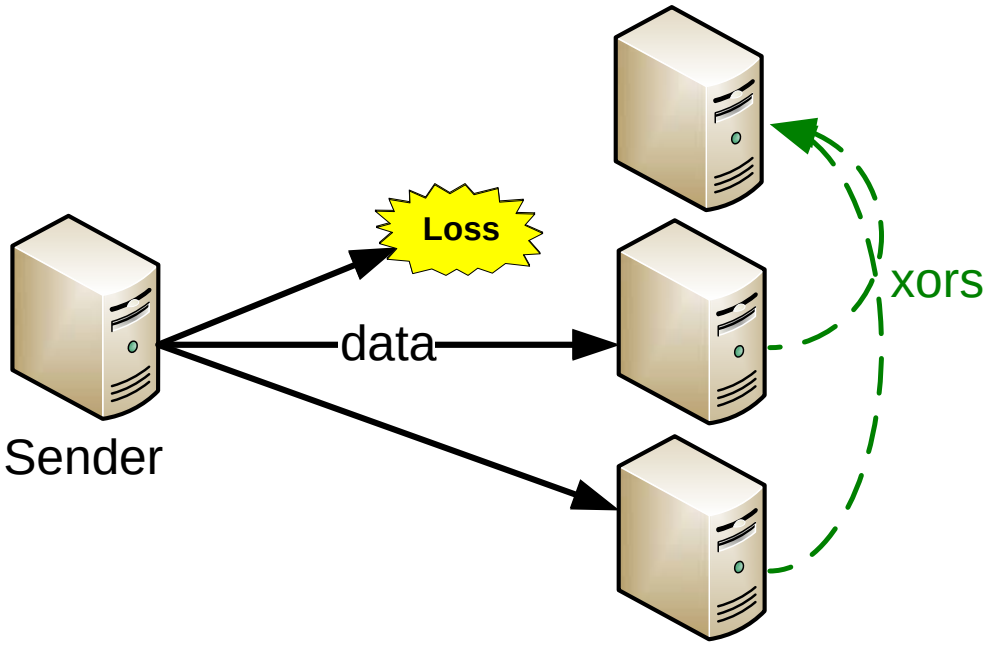
How does latency scale?



- 1. acks: implosion
- 2. naks
- 3. Sender-based FEC
  $\textit{recovery latency} \propto \frac{1}{\textit{datarate}}$
  data rate:
  one sender, one group
- FEC in the network:
  Where and What?

Design Space for Reliable Multicast

How does latency scale?

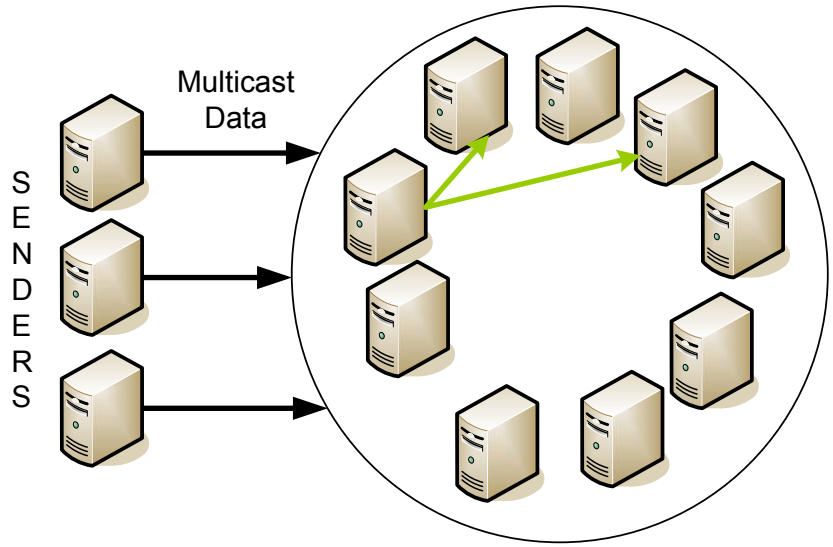


- 1. acks: implosion
- 2. naks
- 3. Sender-based FEC
  recovery latency $\propto \frac{1}{datarate}$
  data rate:
  one sender, one group
- FEC in the network:
  Where and What?

Receiver-based FEC: at receivers, from incoming data

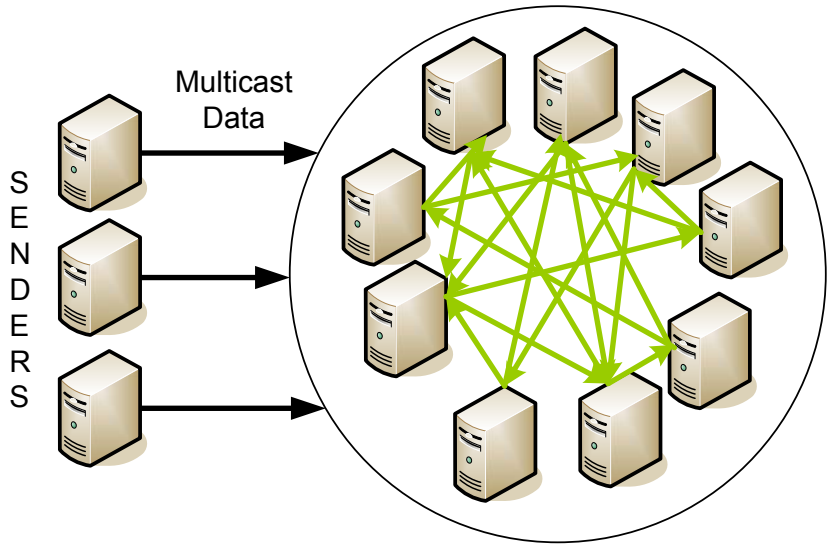
Receiver-Based Forward Error Correction

- Receiver generates an XOR of r incoming multicast packets and exchanges with other receivers
- Each XOR sent to c other random receivers
- Rate: (r, c)

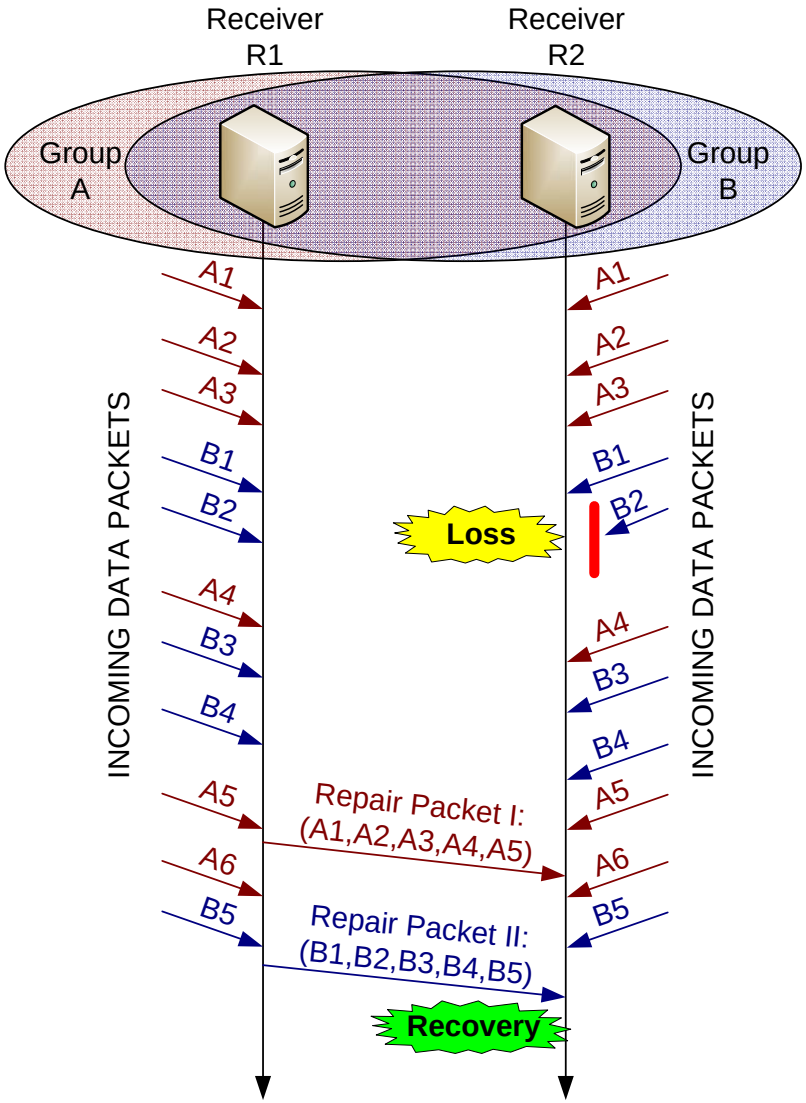


Receiver-Based Forward Error Correction

- Receiver generates an XOR of r incoming multicast packets and exchanges with other receivers
- Each XOR sent to c other random receivers
- Rate: (r, c)
- $latency \propto \frac{1}{\sum_s datarate}$
  data rate: across all senders, in a single group

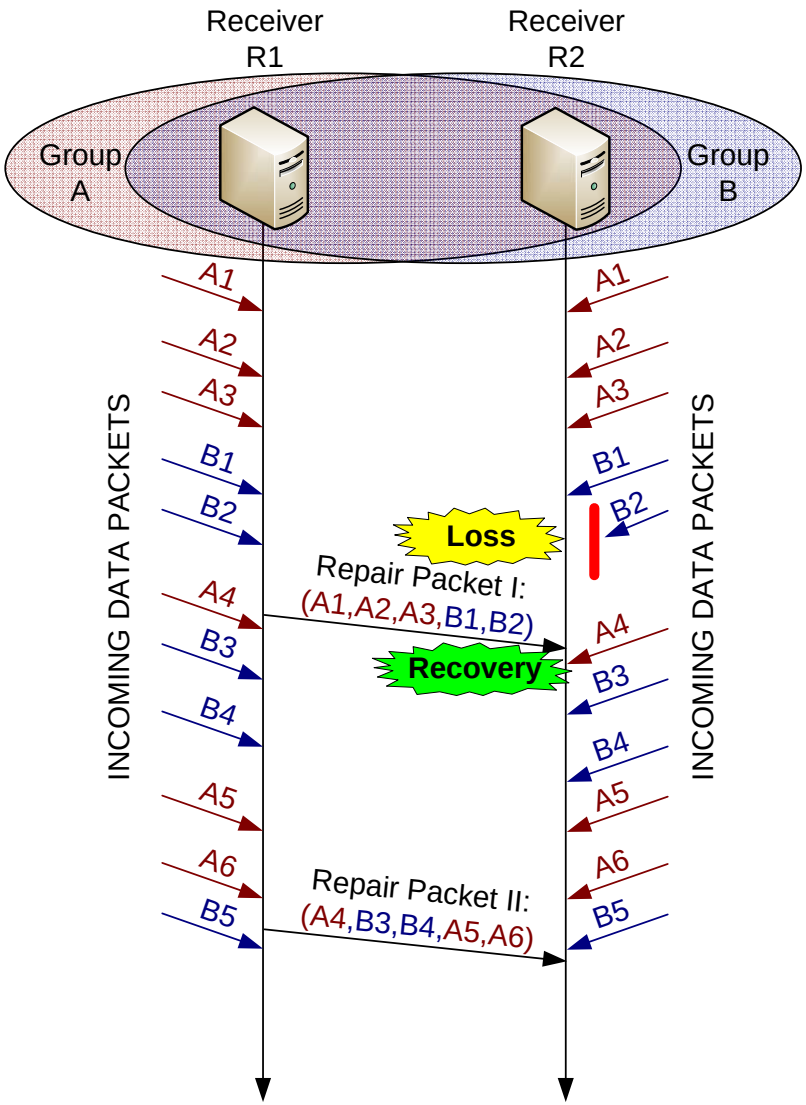


Lateral Error Correction: Principle



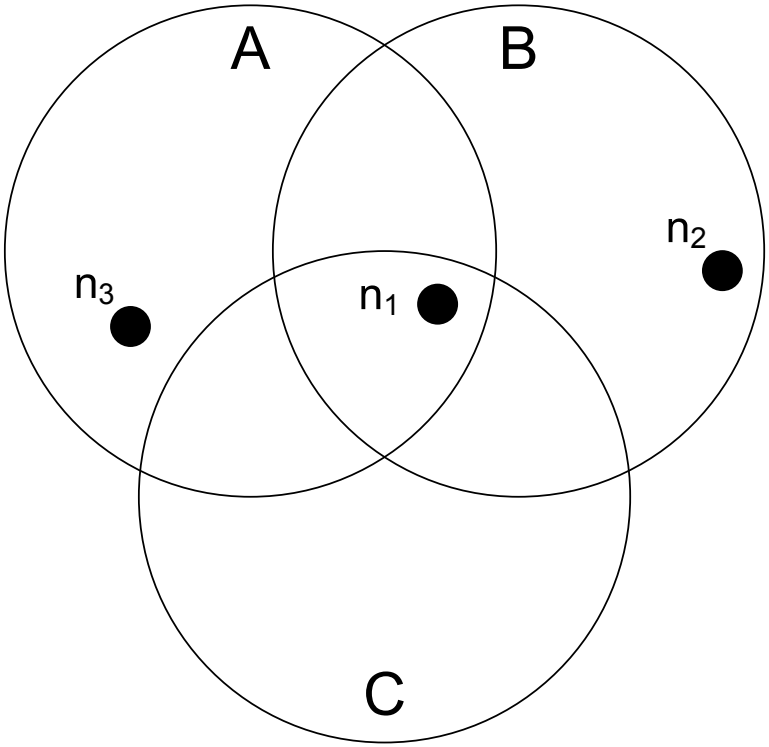
- Single-Group RFEC

Lateral Error Correction: Principle



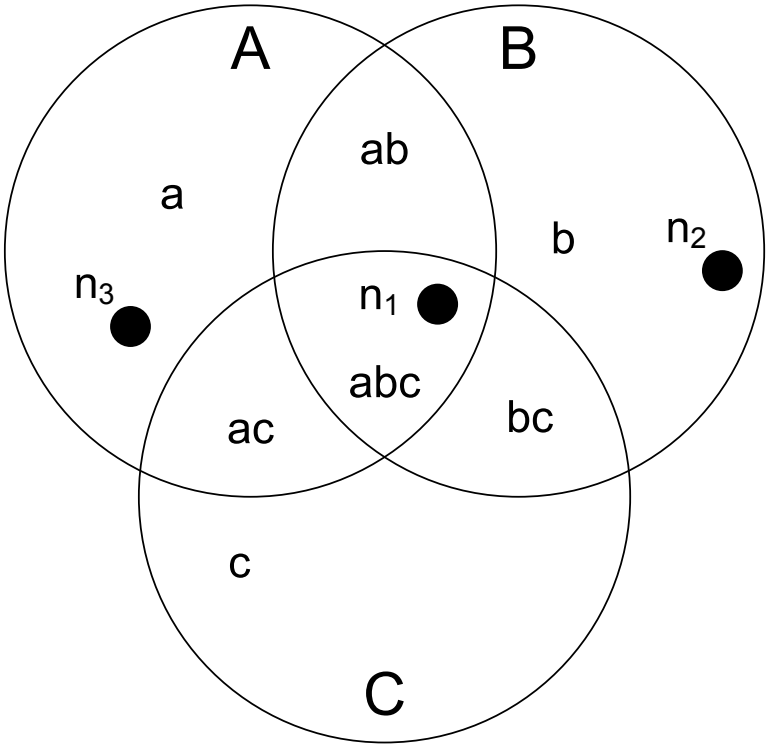
- Single-Group RFEC
- Lateral Error Correction
  - Create XORs from multiple groups → faster recovery!
- What about complex overlap?

Nodes and Disjoint Regions



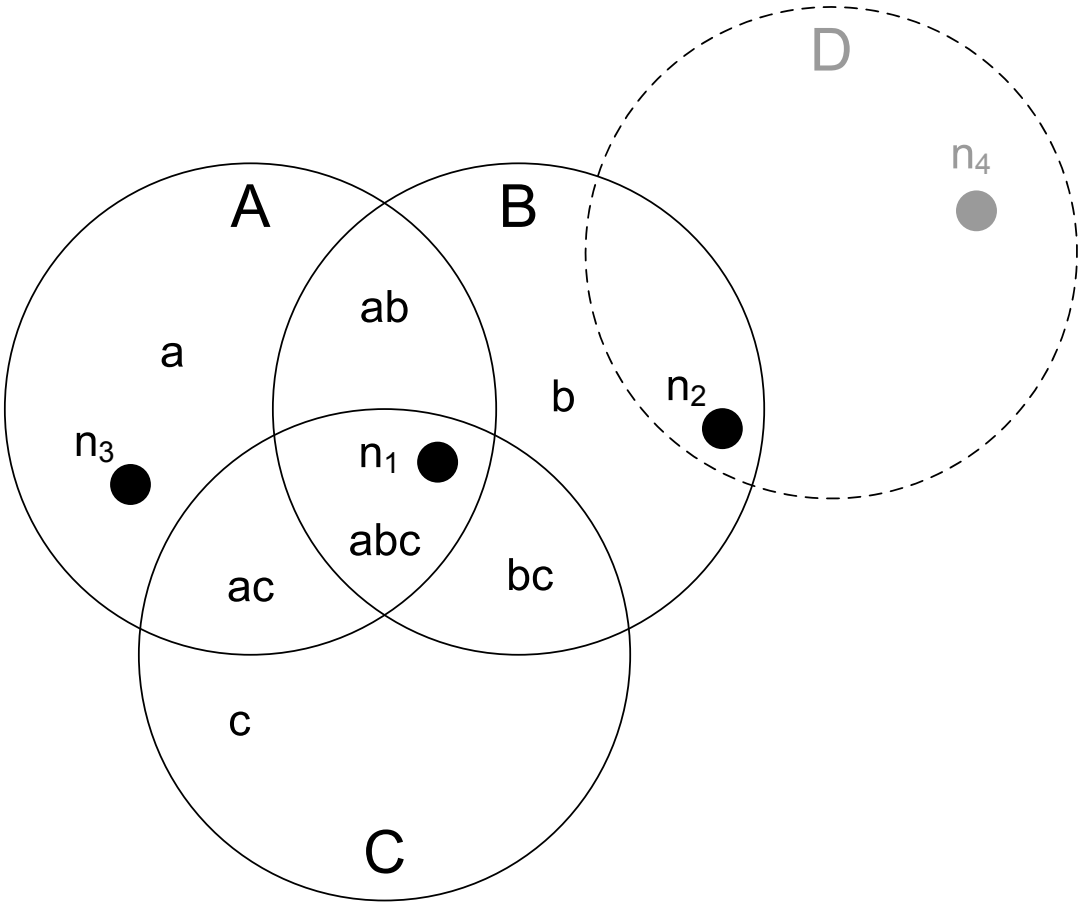
- Receiver n_1 belongs to groups A, B, and C

Nodes and Disjoint Regions



- Receiver n_1 belongs to groups A, B, and C
- Divides groups into disjoint regions

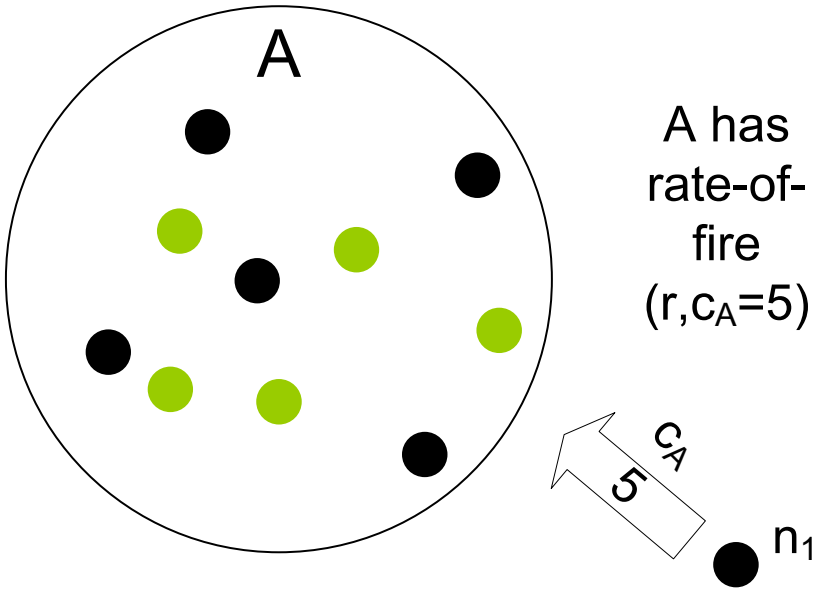
Nodes and Disjoint Regions



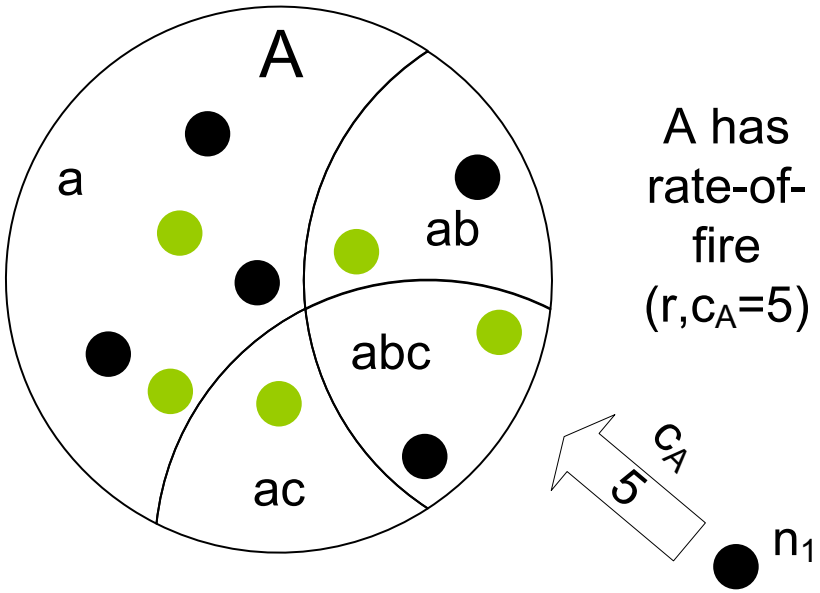
- Receiver n_1 belongs to groups *A*, *B*, and *C*
- Divides groups into disjoint regions
- Is unaware of groups it does not belong to (D)

- Works with any conventional Group Membership Service

Regional Selection

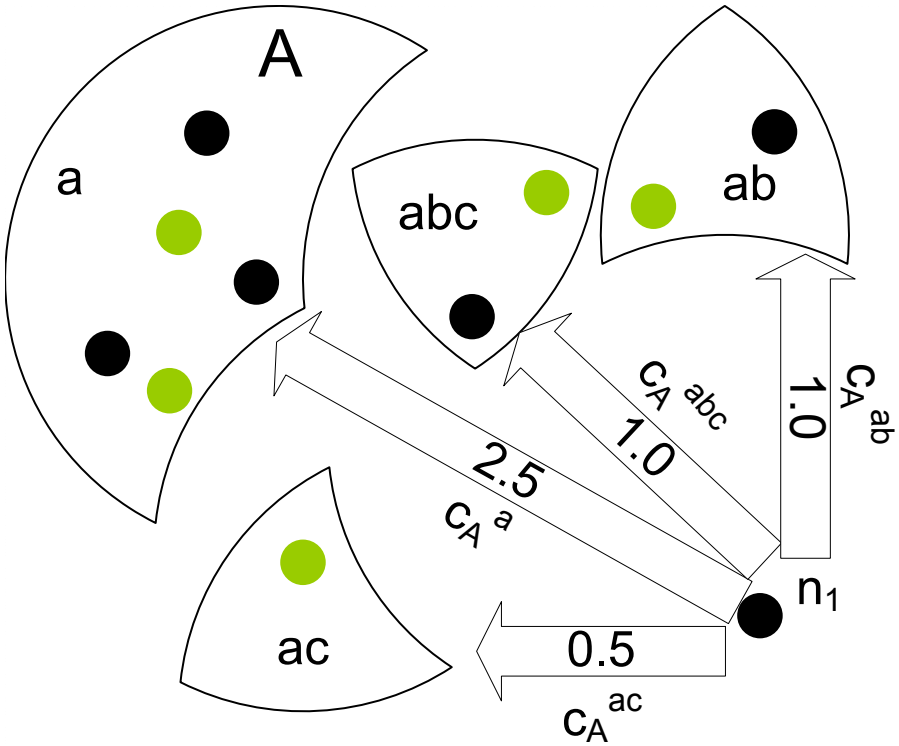


Regional Selection



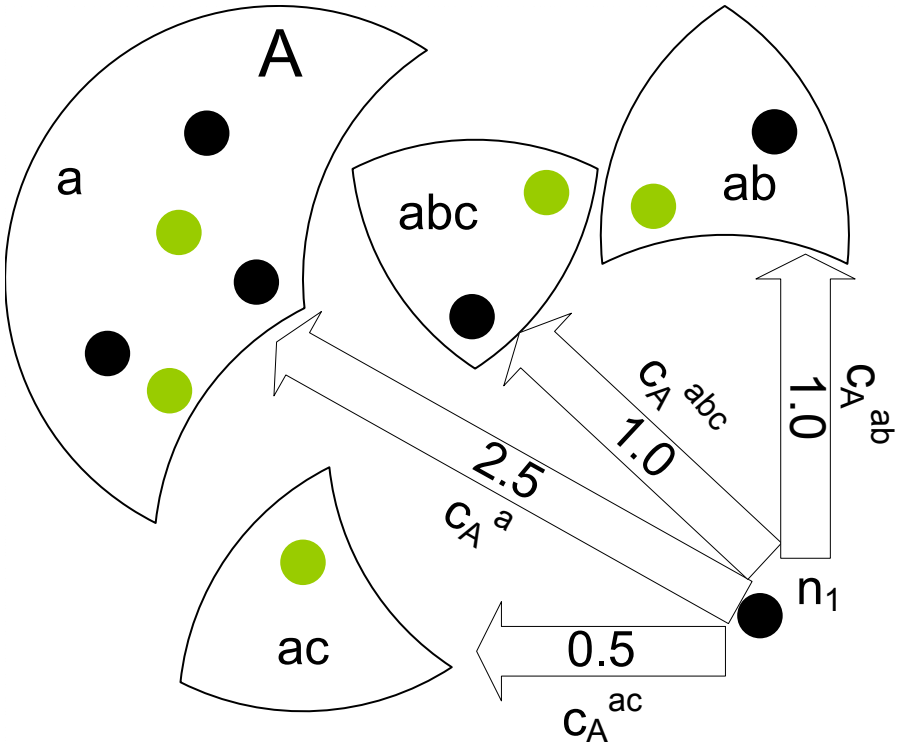
- Select targets for XORs from *regions*, not groups

Regional Selection



- Select targets for XORs from *regions*, not groups
- From each region, select proportional fraction of c_A: $c_A^x = \frac{|x|}{|A|} \cdot c_A$

Regional Selection



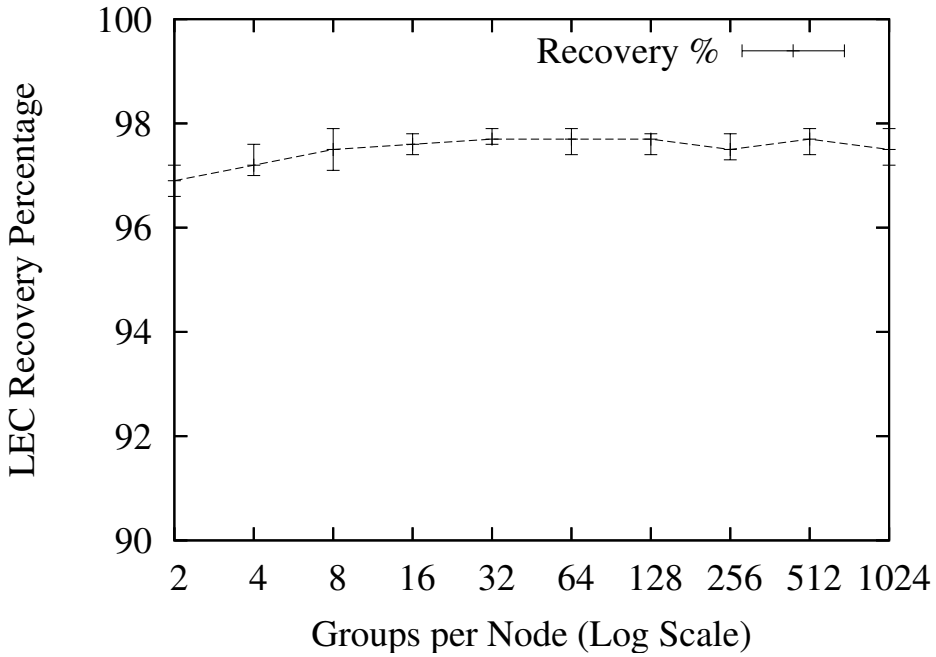
- Select targets for XORs from *regions*, not groups
- From each region, select proportional fraction of c_A: $c_A^x = \frac{|x|}{|A|} \cdot c_A$

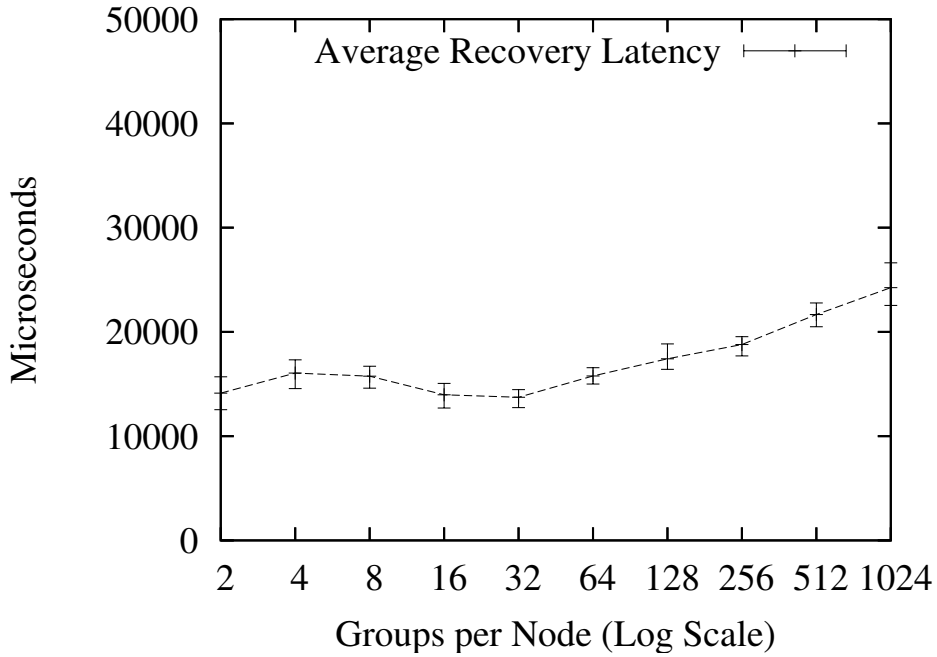
$$latency \propto \frac{1}{\sum_s \sum_g datarate}$$

data rate: across all senders, in intersections of groups

Scalability in Groups

Claim: Ricochet *scales* to hundreds of groups.

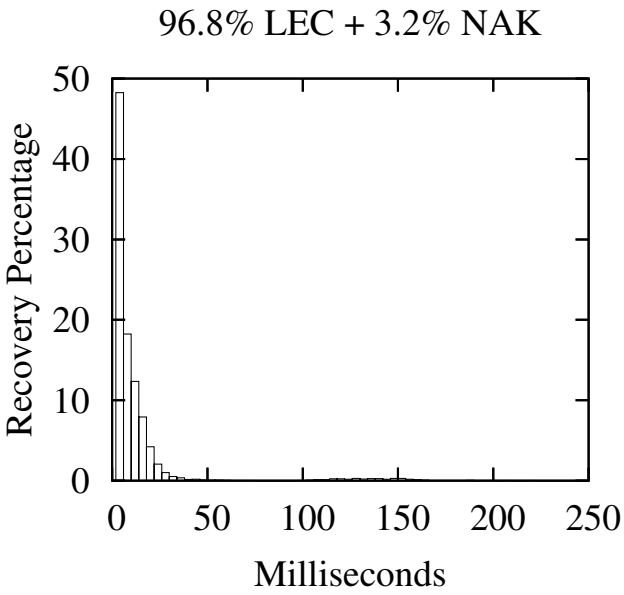




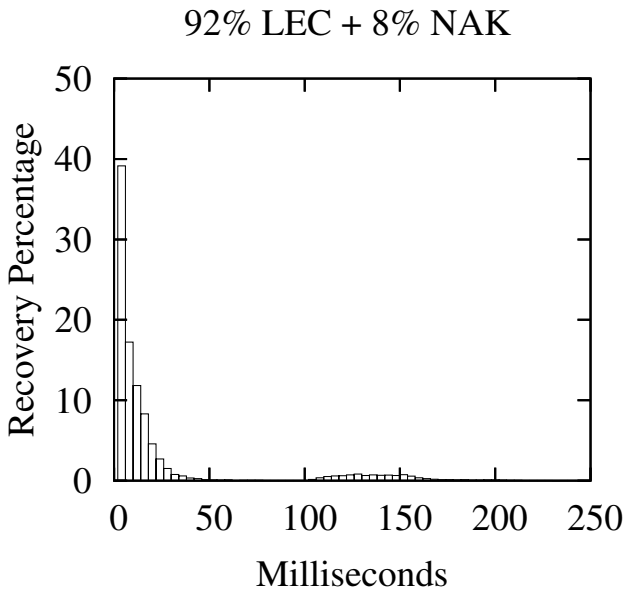
Comparision: At 128 groups, NAK/SFEC latency is 8 seconds.
Ricochet is 400 times faster!

Distribution of Recovery Latency

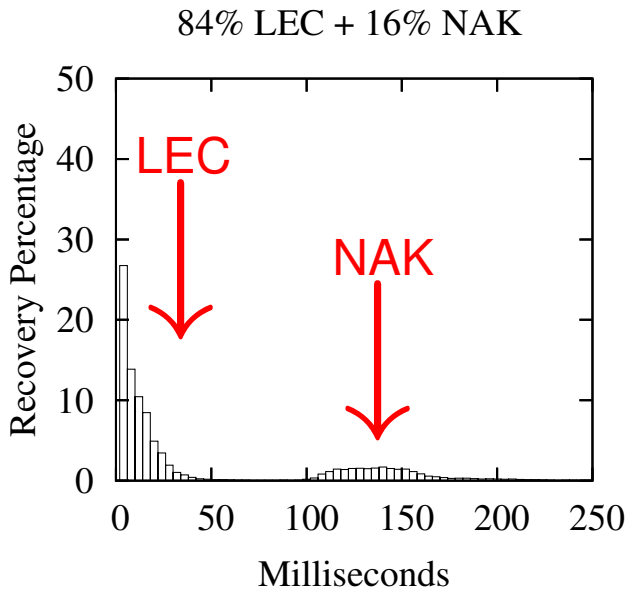
Claim: Ricochet is *reliable* and *time-critical*



(a) 10% Loss Rate



(b) 15% Loss Rate



(c) 20% Loss Rate

Most lost packets recovered $<$ 50ms by LEC.
Remainder via reactive NAKs.

Bursty Loss: 100 packet burst $\rightarrow$ 90% recovered at 50 ms avg

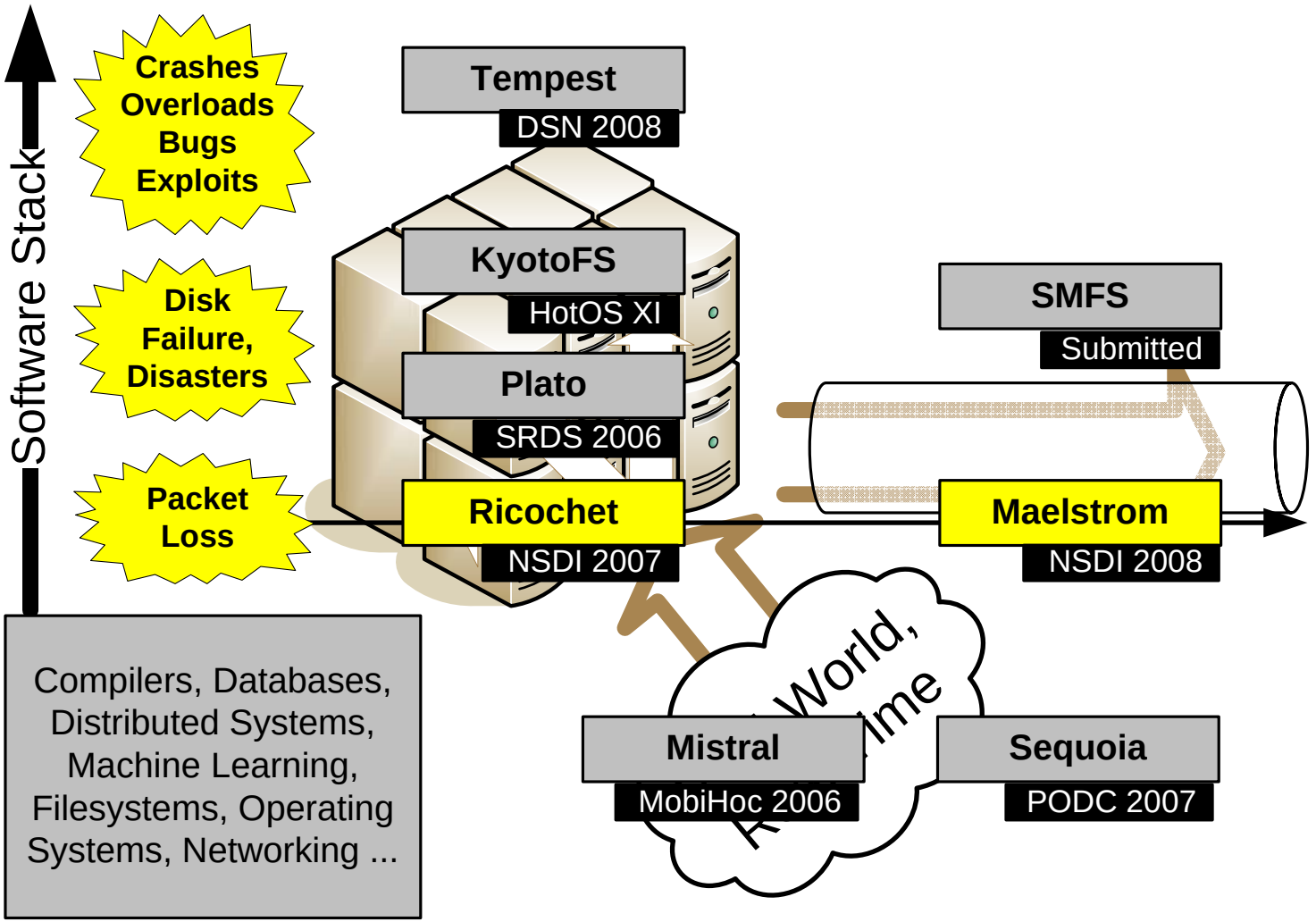
Next Step: Dr. Multicast

- IP Multicast has a bad reputation!
  - Unscalable filtering at routers/switches/NICs
  - Insecure

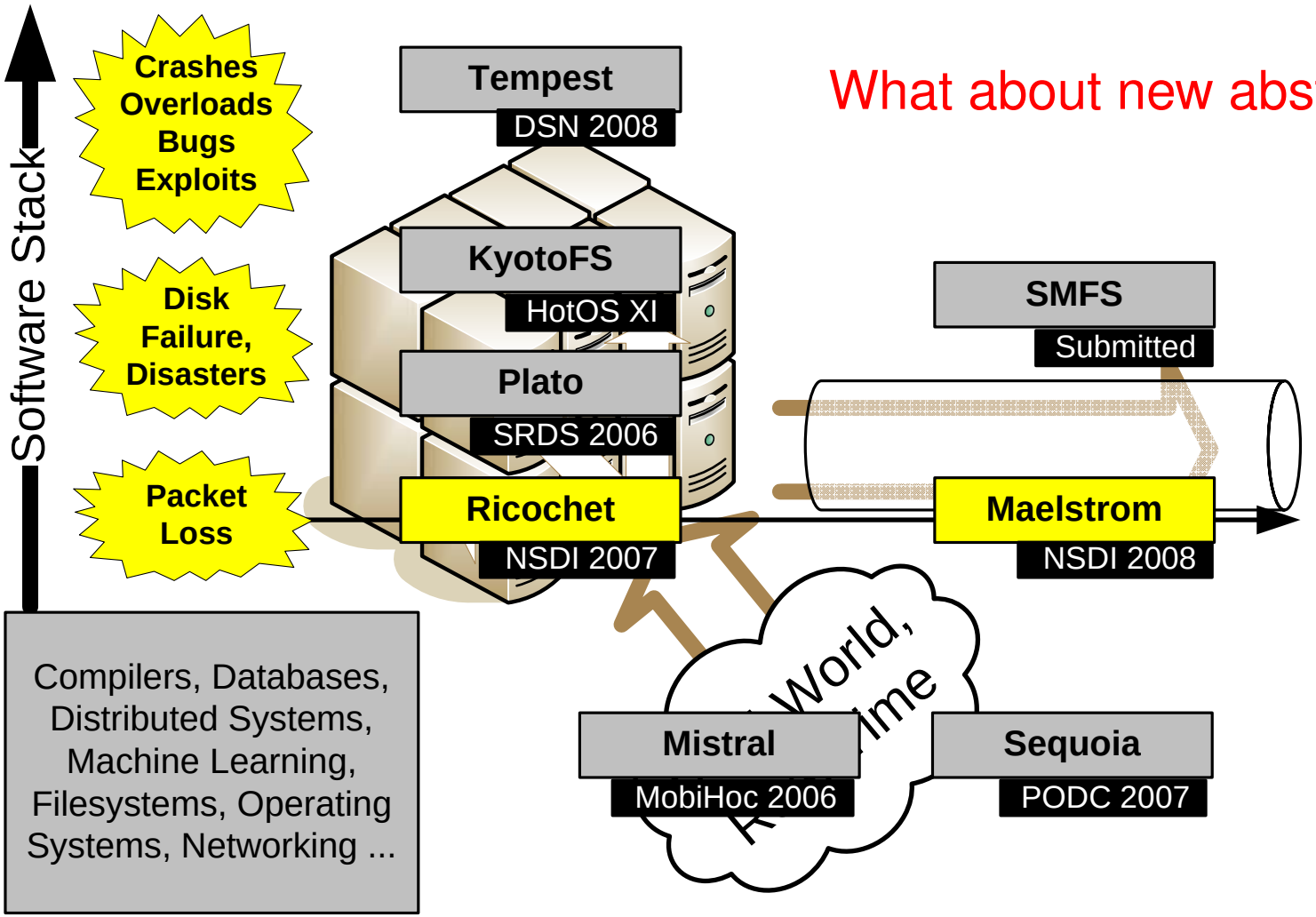
Next Step: Dr. Multicast

- IP Multicast has a bad reputation!
  - Unscalable filtering at routers/switches/NICs
  - Insecure
- Insight: IP Multicast is a shared, controlled resource
  - Transparent interception of socket system calls
  - Logical address $\rightarrow$ Set of network (uni/multi)cast addresses
  - Enforcement of IP Multicast policies
  - Gossip-based tracking of membership/mappings

The Big Picture



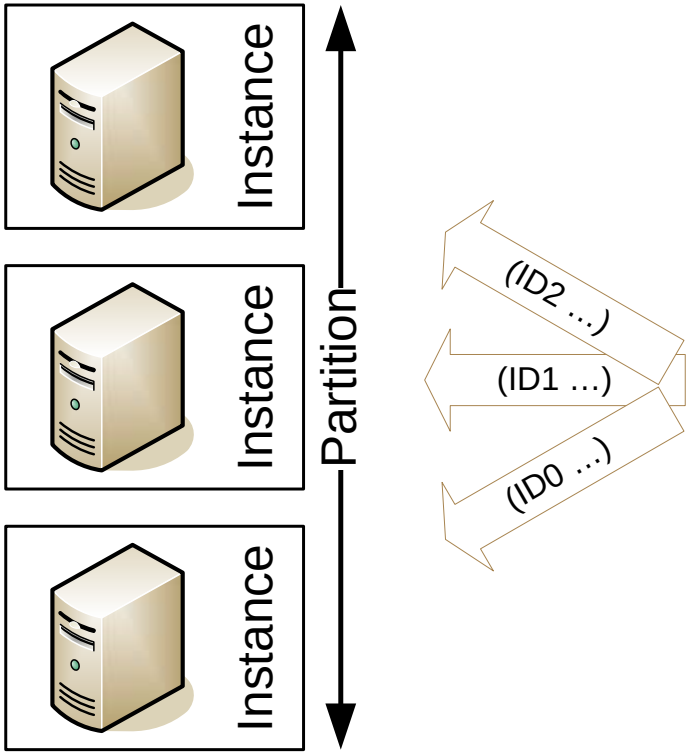
The Big Picture



What about new abstractions?

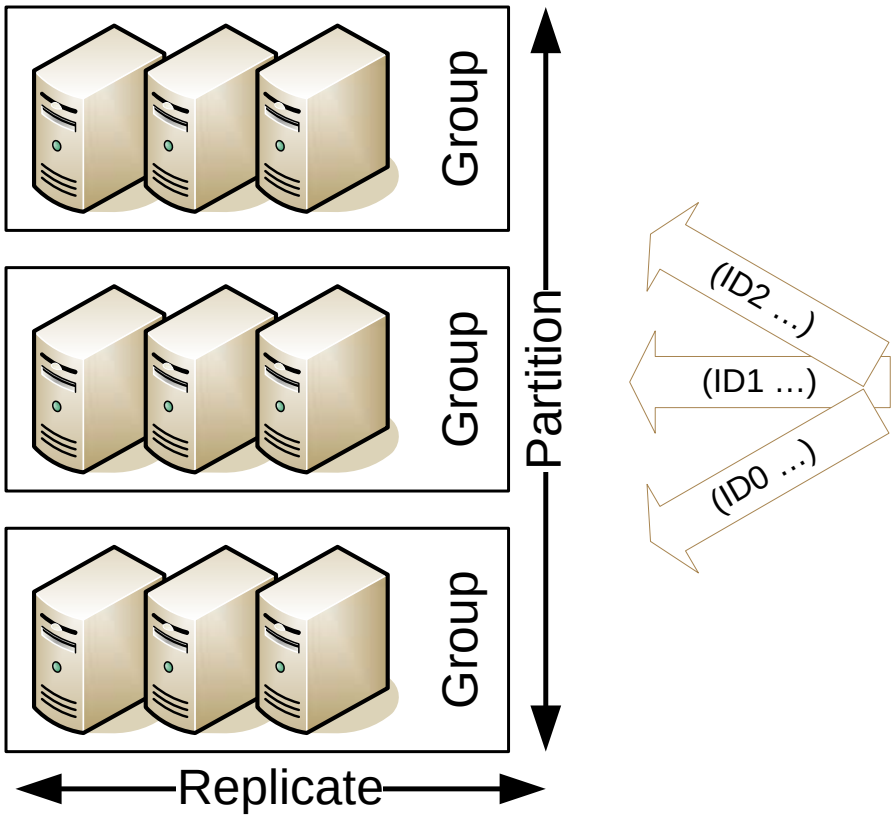
Future Work

Future Work



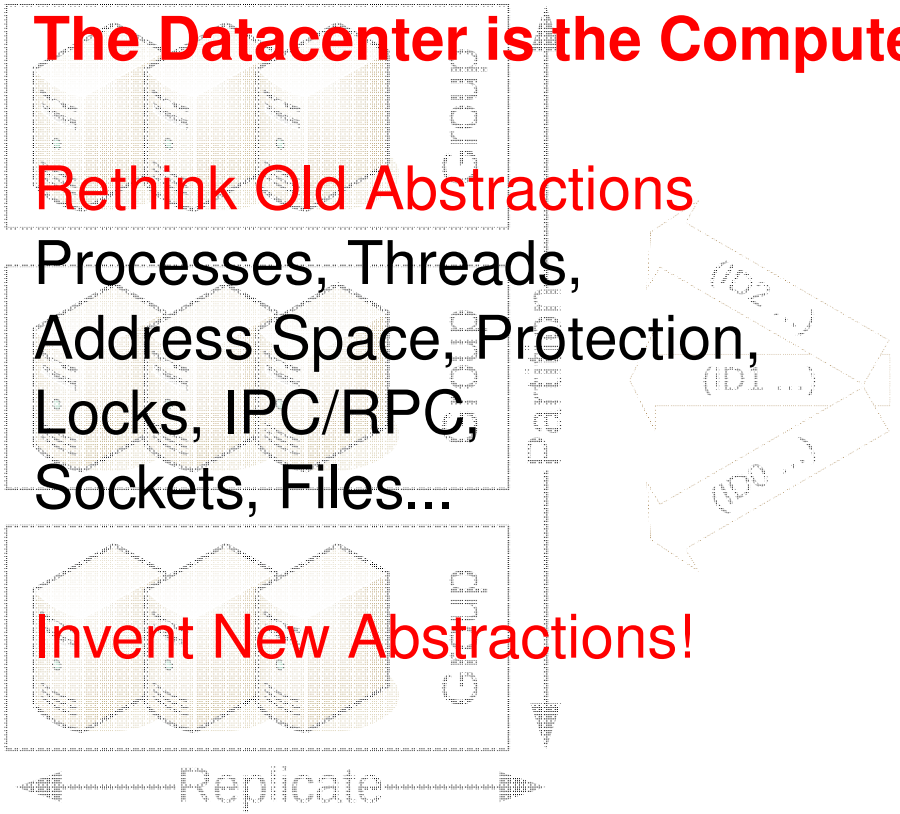
- Partition for Scalability

Future Work



- Partition for Scalability
- Replicate for Availability / Fault-Tolerance

Future Work



- Partition for Scalability
- Replicate for Availability / Fault-Tolerance
- The DatacenterOS

  Concerns: Performance, Parallelism, Privacy, Power...

Conclusion

- The Real-Time Datacenter
  - Recover from failures within seconds
- Reliable Communication: FEC in the Network
  - Recover lost packets in milliseconds
  - Maelstrom: between Datacenters
  - Ricochet: within Datacenters

Conclusion

- The Real-Time Datacenter
  - Recover from failures within seconds
- Reliable Communication: FEC in the Network
  - Recover lost packets in milliseconds
  - Maelstrom: between Datacenters
  - Ricochet: within Datacenters

Thank You!

Extra Slide: FEC and Bursty Loss

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

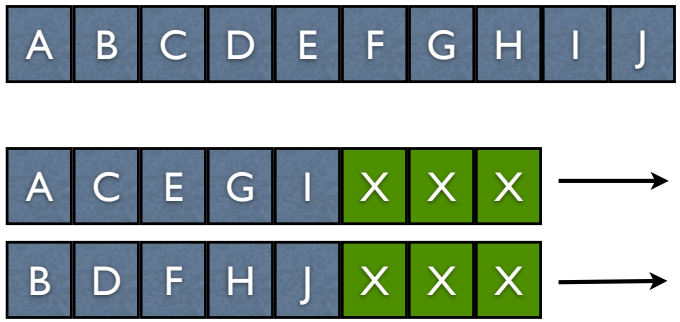


Figure: Interleave of 2 — Even and Odd packets encoded separately

Extra Slide: FEC and Bursty Loss

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

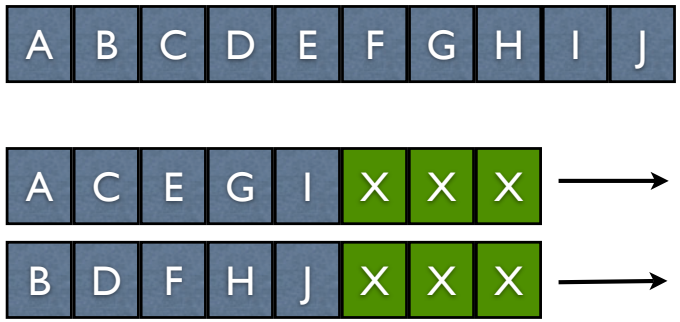
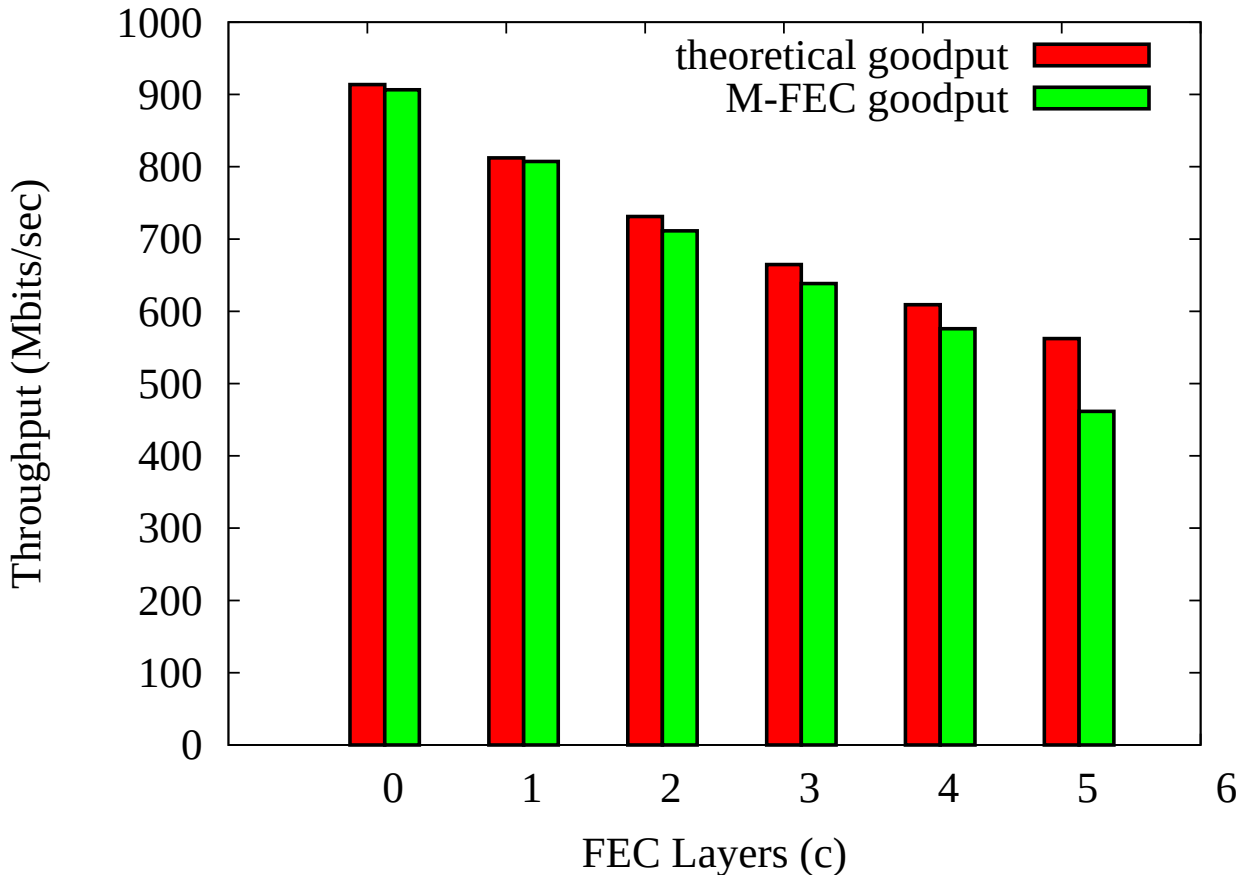


Figure: Interleave of 2 — Even and Odd packets encoded separately

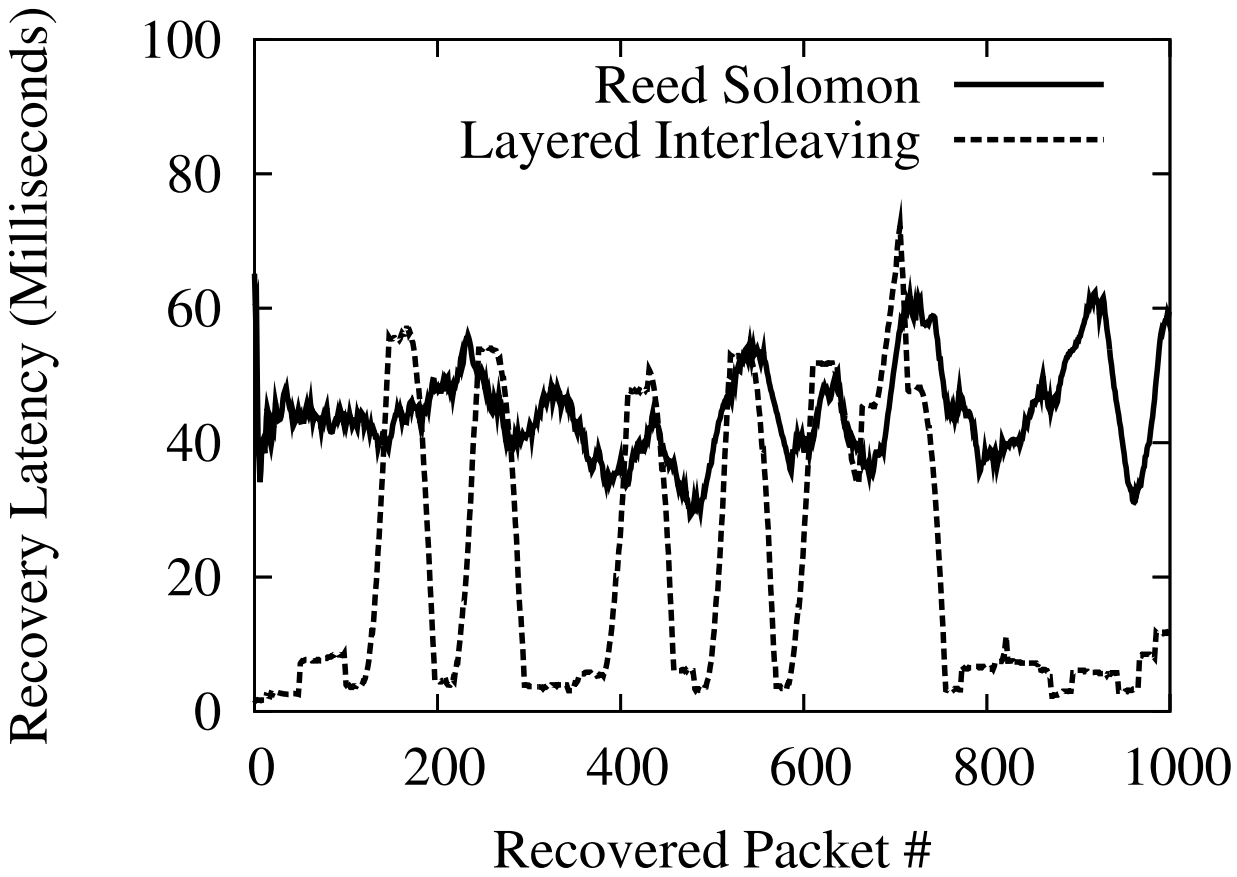
Wanted: Graceful degradation of recovery latency with actual burst size for constant overhead

Extra Slide: Maelstrom Evaluation

Maelstrom goodput is near theoretical maximum



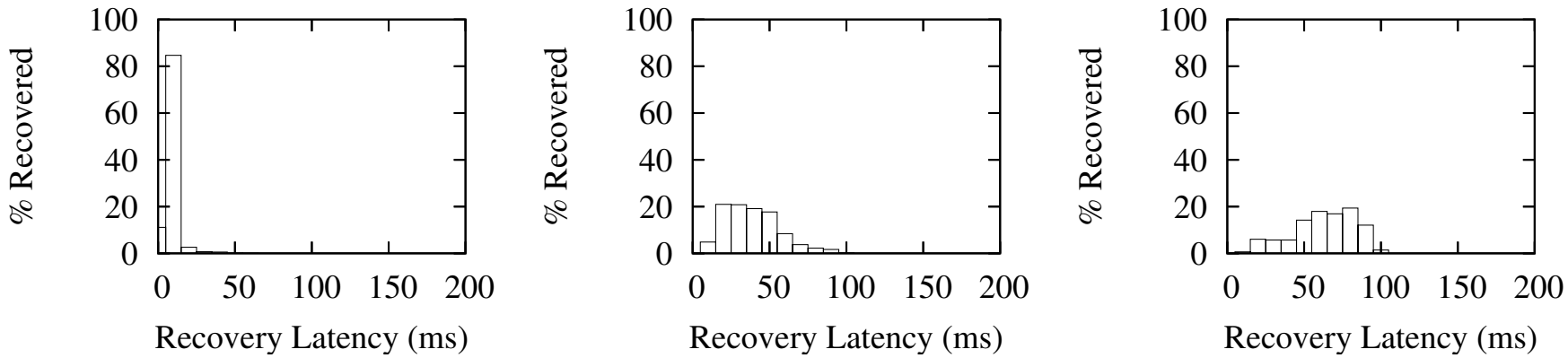
Extra Slide: Layered Interleaving



Evaluation: Layered Interleaving

Claim: Recovery Latency depends on Actual Burst Length

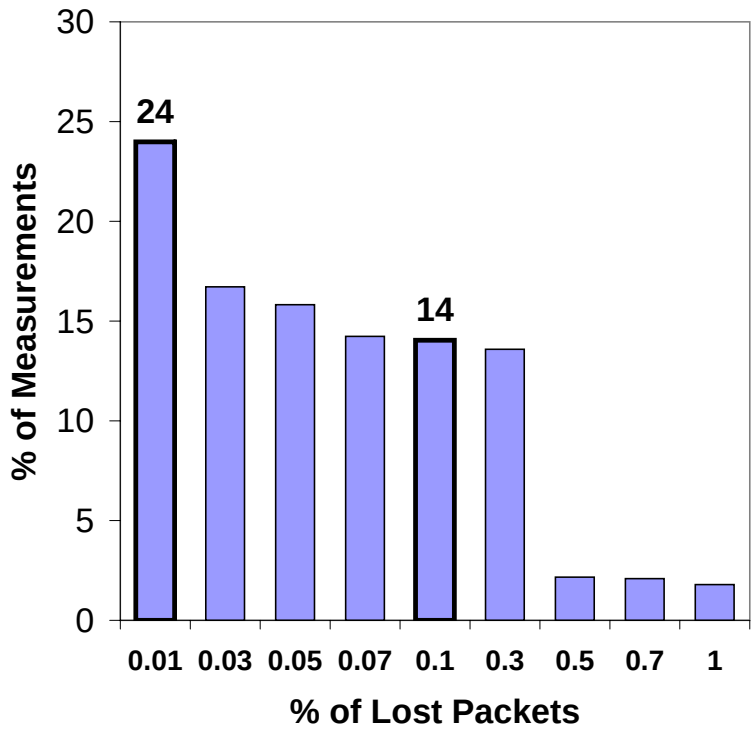
Burst Length = 1 20 40



- Longer Burst Lengths → Longer Recovery Latency

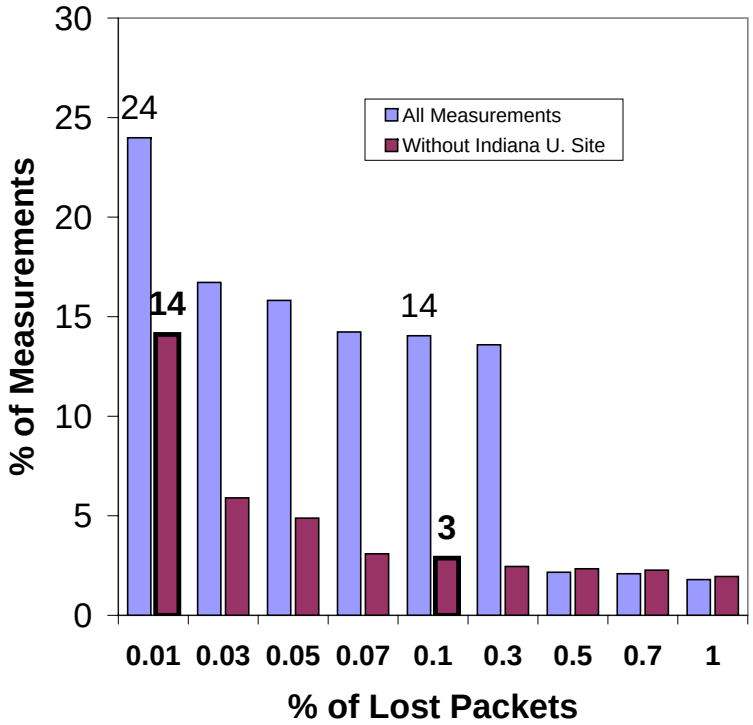
TeraGrid: Supercomputer Network

- 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
  - ...



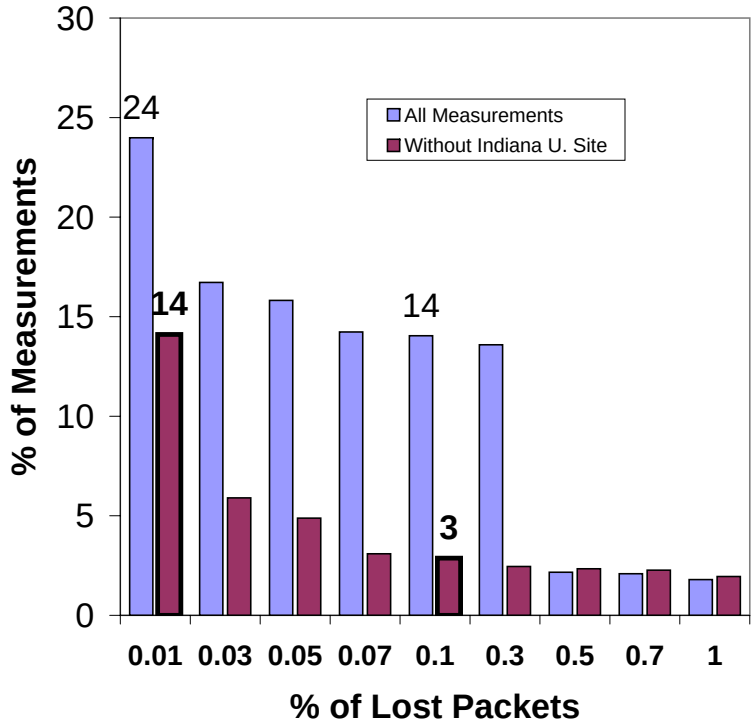
TeraGrid: Supercomputer Network

- 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
  - ...



TeraGrid: Supercomputer Network

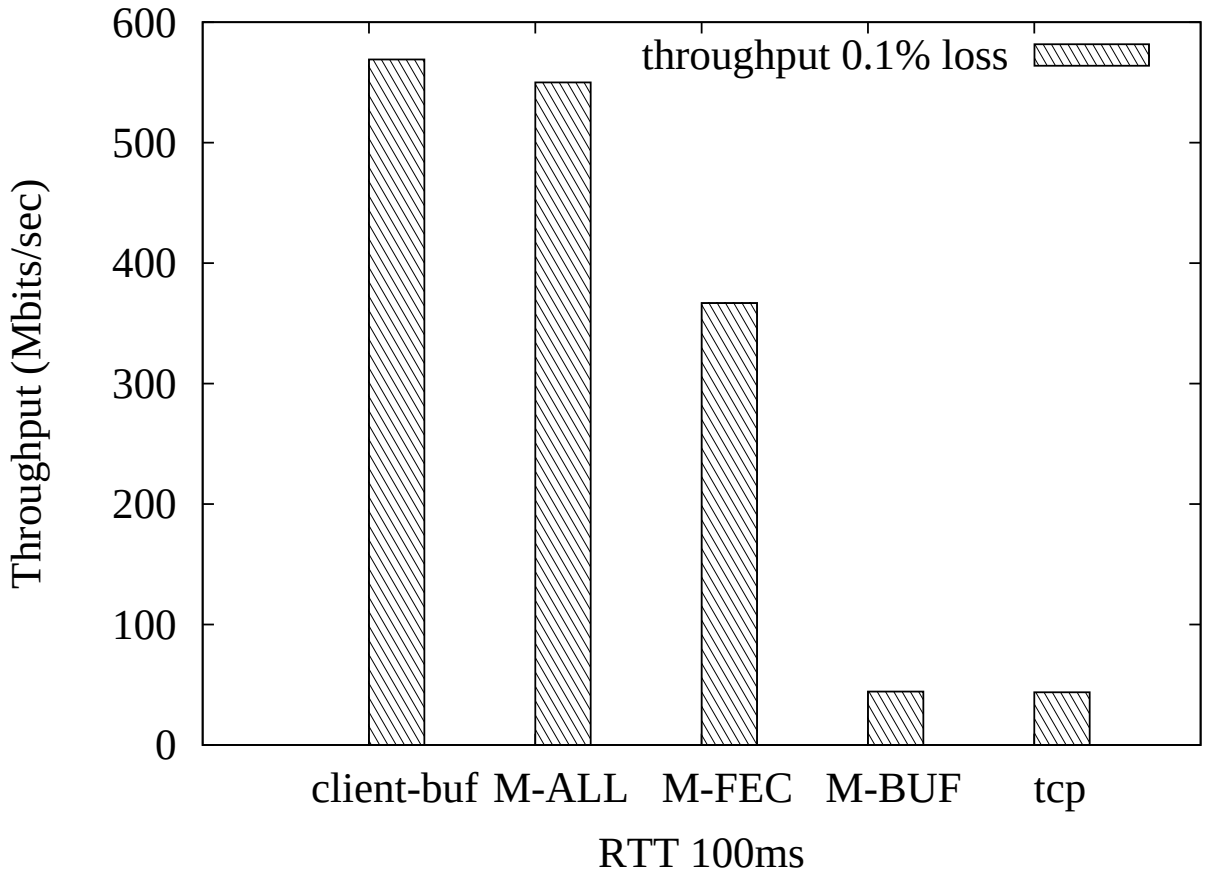
- End-to-End UDP Probes: Zero Congestion, Non-Zero Loss!
- Possible Reasons:
  - transient congestion
  - degraded fiber
  - malfunctioning HW
  - misconfigured HW
  - switching contention
  - low receiver power
  - end-host overflow
  - ...



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

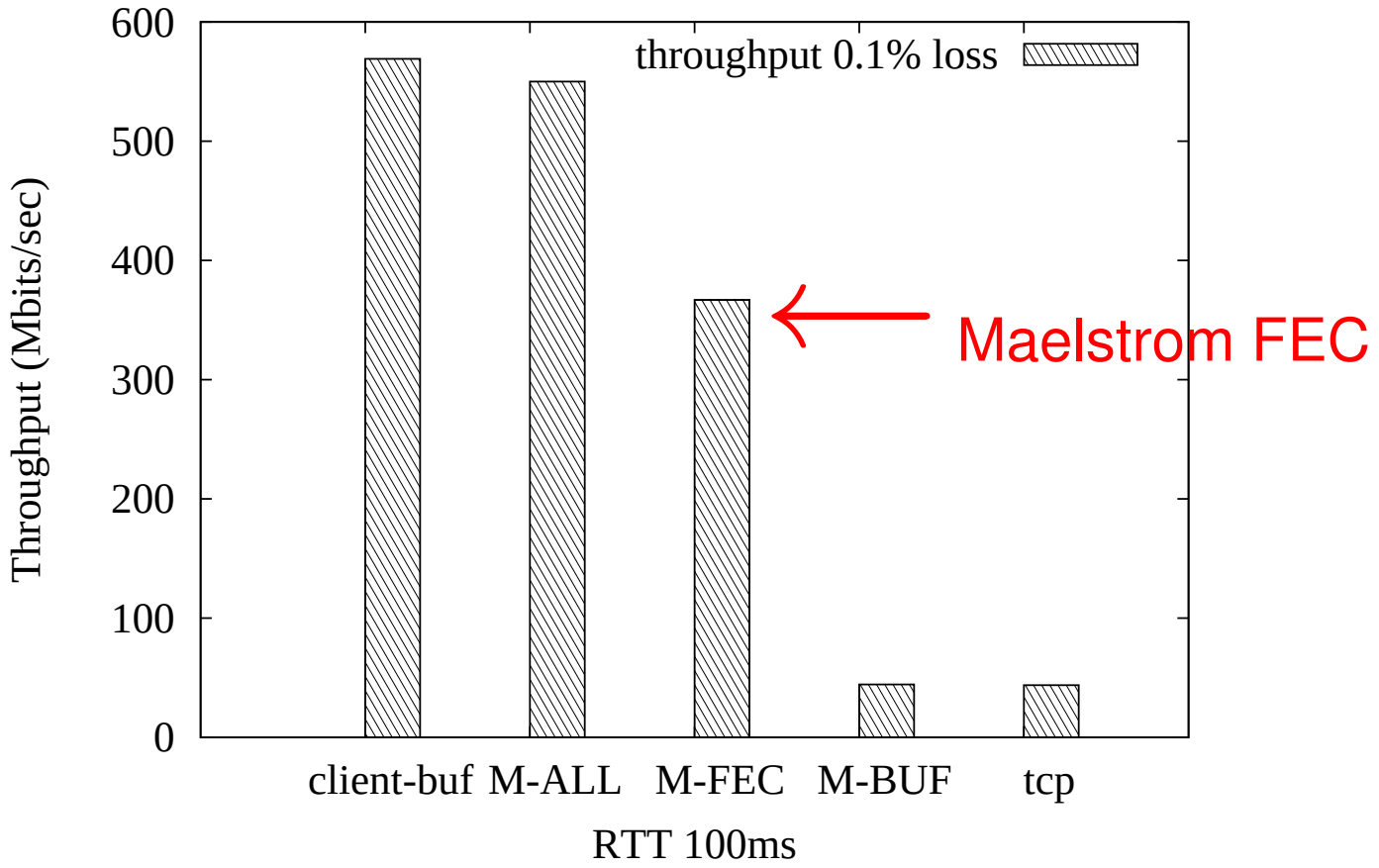
Evaluation: Split Mode and Buffering

Claim: Maelstrom eliminates the need for large end-host buffers



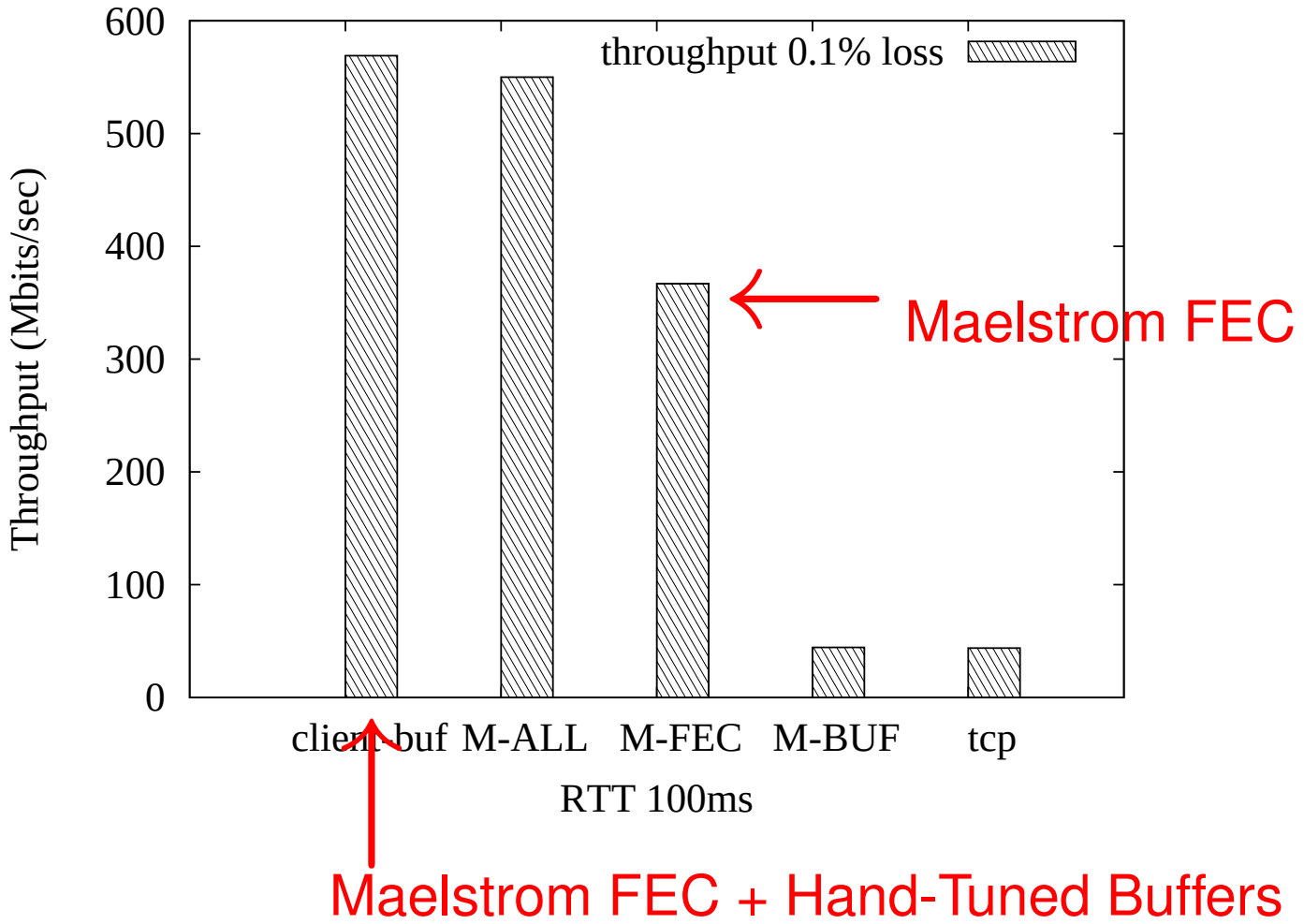
Evaluation: Split Mode and Buffering

Claim: Maelstrom eliminates the need for large end-host buffers



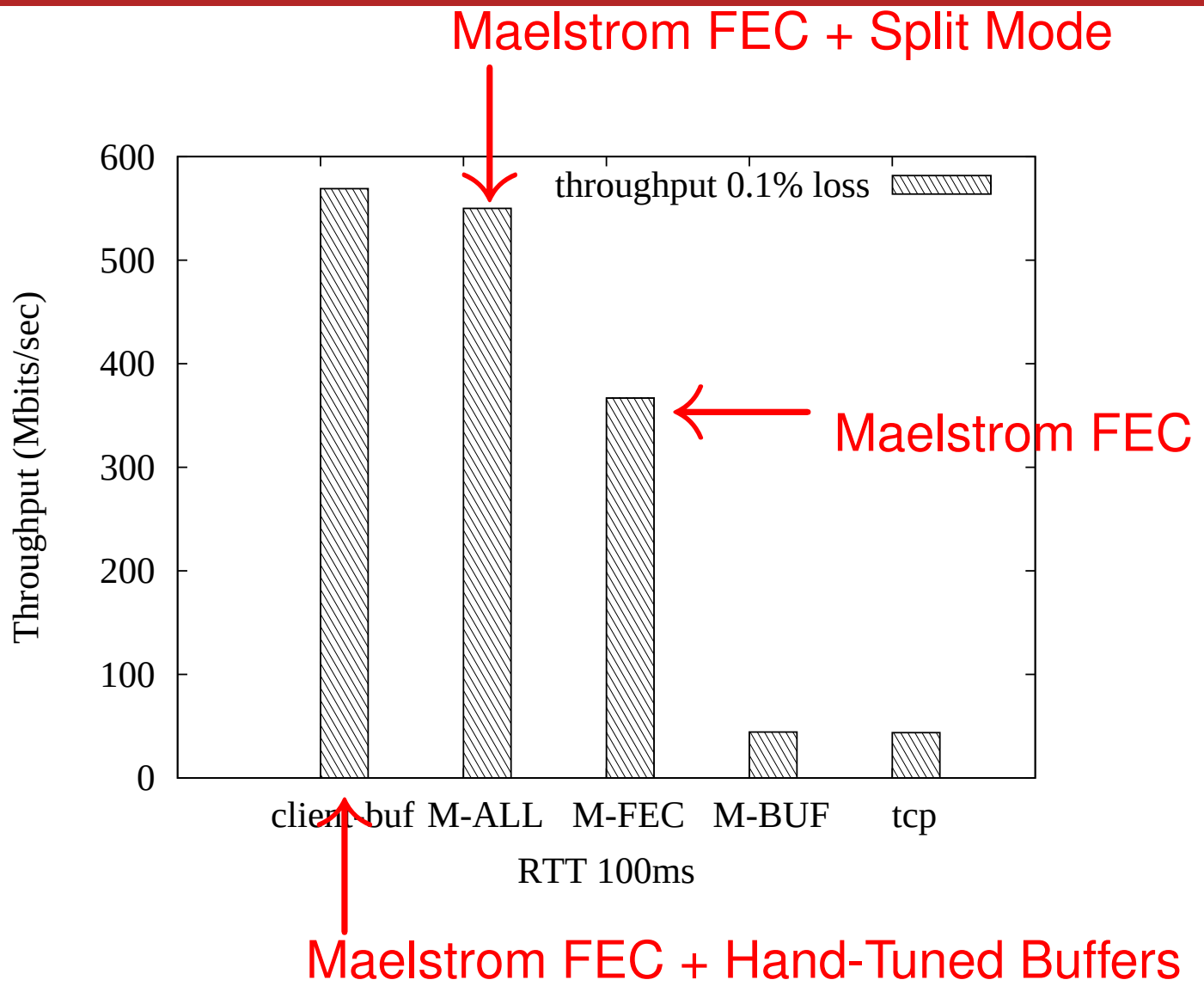
Evaluation: Split Mode and Buffering

Claim: Maelstrom eliminates the need for large end-host buffers



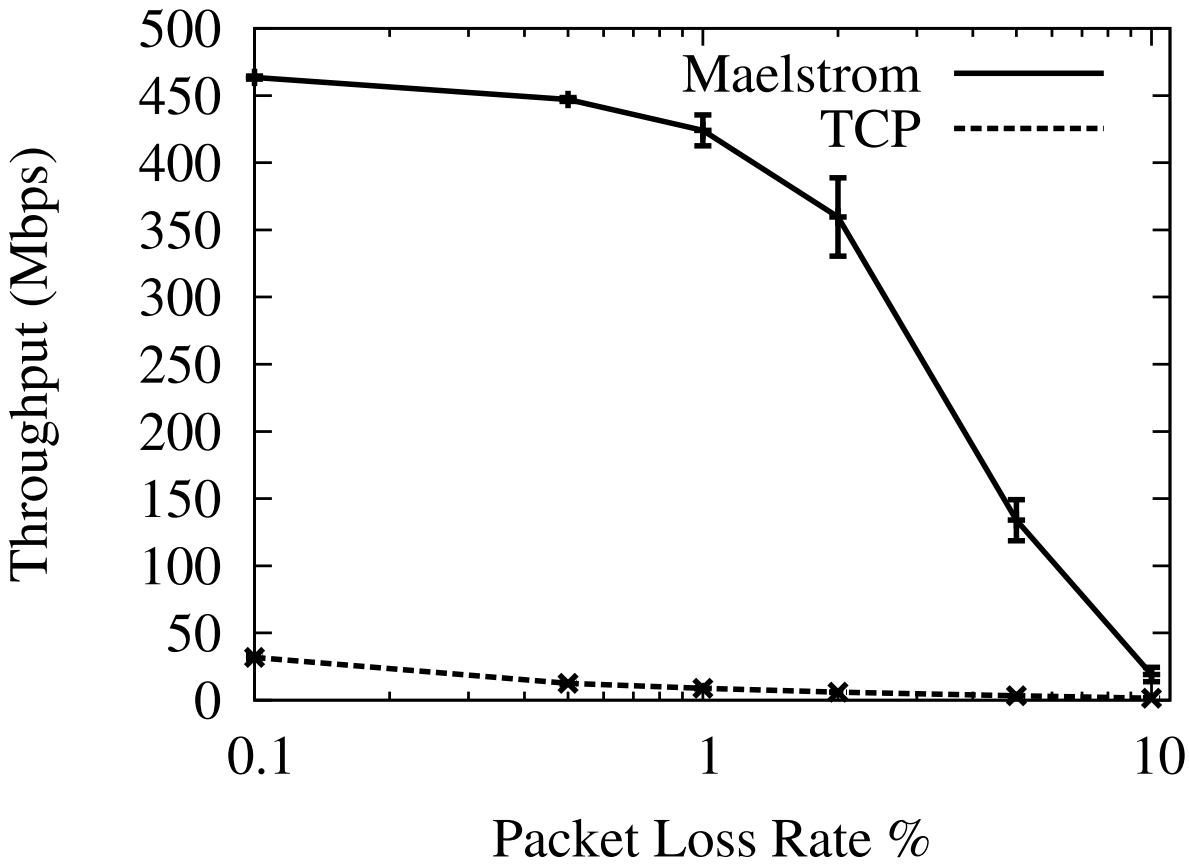
Evaluation: Split Mode and Buffering

Claim: Maelstrom eliminates the need for large end-host buffers



Evaluation: FEC mode and loss

Claim: Maelstrom works at high loss rates



Link RTT = 100ms