Ricochet: Lateral Error Correction for Time-Critical Multicast

Mahesh Balakrishnan\textsuperscript{1}, Ken Birman\textsuperscript{1}, Amar Phanishayee\textsuperscript{2}, Stefan Pleisch\textsuperscript{1}

\textsuperscript{1}Cornell University, Ithaca, NY
\textsuperscript{2}Carnegie Mellon University, Pittsburgh, PA
Multicast in the Datacenter

- Commodity Datacenters: Extreme Scale-Out
- Data Replication:
  - Fault Tolerance
  - High Availability
  - Performance
Multicast in the Datacenter

- Commodity Datacenters: Extreme Scale-Out
- Data Replication:
  - Fault Tolerance
  - High Availability
  - Performance
- Updating data in multiple locations: Multicast!
How is Multicast Used?

Financial Pub-Sub Example:
- Each equity is mapped to a multicast group.
- Each node is interested in a different set of equities ...

![Diagram with nodes and multicast groups](image-url)
How is Multicast Used?

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 ⇒ Low per-group data rate
How is Multicast Used?

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

⇒ Low per-group data rate

High per-node data rate

⇒ Overload
Recovering Lost Packets... Fast!

- Loss occurs on overloaded end-hosts.
- Real-Time Apps: Financial Trading, Mission Control...
- Foobooks.com?
  - Massive volume...
  - Stale inventory = Lost $$$
- Required: rapid, scalable recovery from packet loss
Problem Statement

- Time-Critical Reliable Multicast
- Scalability:

[Diagram showing a sender sending multicast data to a group, with one packet marked as lost.]
Problem Statement

- Time-Critical Reliable Multicast
- Scalability:
  - Number of Receivers
Problem Statement

- Time-Critical Reliable Multicast
- Scalability:
  - Number of Receivers
  - Number of Senders
Problem Statement

- Time-Critical Reliable Multicast
- Scalability:
  - Number of Receivers
  - Number of Senders
  - Number of Groups
Design Space for Reliable Multicast
How does latency scale?

Existing mechanisms:
- ACK/timeout: RMTP/RMTP-II
- NAK/sender-based sequencing: SRM
- Gossip-based: Bimodal Multicast, lpbcast
- Forward Error Correction

Fundamental Insight: \( \text{latency} \propto \frac{1}{\text{datarate}} \)
NAK/Sender-Based Sequencing: SRM

Scalable Reliable Multicast - Developed 1997

- Loss *discovered* on next packet from same sender in same group
- *latency* $\propto \frac{1}{\text{datarate}}$

Data rate: at a single sender, in a single group
Pros:
- Tunable, Proactive Overhead: \((r, c)\)
- *Time-Critical*: No Retransmission

Cons:
- FEC packets are generated over a stream of data
  - Have to wait for \(r\) data packets before generating FEC
  - \(\text{latency} \propto \frac{1}{\text{datarate}}\)

Data rate: at a single sender, in a single group
Receiver-Based Forward Error Correction

- Receivers generate XORs of incoming multicast packets ...

![Diagram of Receiver-Based FEC](image-url)
- Receivers generate XORs of incoming multicast packets ...
- ... and exchange with other receivers
- A receiver can recover from at most one missing packet in an XOR
Receiver-Based Forward Error Correction

- Motivation
- System
- Evaluation
- Design Space
- Receiver-Based FEC
- Lateral Error Correction

Receiver sends XOR to randomly chosen receivers

Gossip-style Randomness

Tunable Overhead: $(r, c)$

Rate-of-fire $\propto \frac{1}{P_s\text{datarate}}$

Data rate: across all senders, in a single group

Mahesh Balakrishnan

Ricochet: Lateral Error Correction for Time-Critical Multicast
• Receiver sends XOR to $c$ randomly chosen receivers
Receiver-Based Forward Error Correction

- Receiver sends XOR to $c$ randomly chosen receivers
- Gossip-style Randomness
- Tunable Overhead: $(r, c)$

*rate-of-fire*
Ricochet: Lateral Error Correction for Time-Critical Multicast

- Receiver sends XOR to $c$ randomly chosen receivers
- Gossip-style Randomness
- Tunable Overhead: $(r, c)$ rate-of-fire
- $latex\text{latency} \propto \frac{1}{\sum_s \text{datarate}}$ data rate: across all senders, in a single group
Lateral Error Correction: Principle

Node \( n_1 \):
- A1
- A2
- A3
- B1
- B2
- A4
- B3
- A5
- A6
- B4
- B5

Node \( n_2 \):
- A1
- A2
- A3
- B1
- B2
- A4
- B3
- A5
- A6
- B4
- B5

INCOMING DATA PACKETS

Repair Packet I: \((A1, A2, A3, A4, A5)\)
Repair Packet II: \((B1, B2, B3, B4, B5)\)

Loss

Recovery

Repairs from \( n_1 \) to \( n_2 \) include data from common groups.

Mahesh Balakrishnan
Ricochet: Lateral Error Correction for Time-Critical Multicast
Lateral Error Correction: Principle

- Repairs from $n_1$ to $n_2$ include data from common groups.

Mahesh Balakrishnan  
Ricochet: Lateral Error Correction for Time-Critical Multicast
Nodes and Intersections

Node $n_1$ belongs to groups $A$, $B$, and $C$
Nodes and Intersections

- Node $n_1$ belongs to groups $A$, $B$, and $C$
- Divides groups into disjoint intersections

Nodes and Intersections
Nodes and Intersections

- Node $n_1$ belongs to groups $A$, $B$, and $C$
- Divides groups into disjoint intersections
- Is unaware of groups it does not belong to (D)
Regional Selection

A has rate-of-fire \((r,c_A=5)\)

\[ n_1 \]

Select targets for repairs from intersections, not groups. From each intersection, select a proportional fraction of \(c_A\):

\[ c_A \cdot |x| \]

Latency \(\propto \frac{1}{P_s P_g}\) data rate: across all senders, in intersections of groups.
Regional Selection

- Select targets for repairs from *intersections*, not groups

A has rate-of-fire \((r,c_A=5)\)
Regional Selection

- Select targets for repairs from *intersections*, not groups.
- From each intersection, select proportional fraction of $c_A$:
  
  $c_A^x = \frac{|x|}{|A|} \cdot c_A$
Regional Selection

Select targets for repairs from *intersections*, not groups.

From each intersection, select proportional fraction of $c_A$:

$$c_A^x = \frac{|x|}{|A|} \cdot c_A$$

Latency $\propto \frac{1}{\sum_s \sum_g \text{datarate}}$

data rate: across all senders, in intersections of groups
Overheads:

- Membership State:
  # of intersections < # of known nodes.
- Computational:
  XORs are fast... 150-300 ms per packet.
- Bandwidth:
  \((r, c) \implies \frac{c}{r+c}\) repair overhead.

- Group Membership Service: Any old one works.
Experimental Evaluation

- Cornell Cluster: 64 1.3 Ghz nodes
- Java Implementation running on Linux 2.6.12
- Three Loss Models: \{Uniform, Burst, Markov\}
- Grouping Parameters: \( g \times s = d \times n \)
  - \( g \): Number of Groups in System
  - \( s \): Average Size of Group
  - \( d \): Groups joined by each Node
  - \( n \): Number of Nodes in System
- Each node joins \( d \) randomly selected groups from \( g \) groups
Where does loss occur in a Datacenter?

Packet Loss occurs at end-hosts: **independent and bursty**

- **Overloaded Node**
- **Lightly Loaded Node**

Traffic Bursts
Loss Bursts
Motivation
System
Evaluation

Distribution of Recovery Latency
16 Nodes, 128 groups per node, 10 nodes per group, Uniform *% Loss

96.8% LEC + 3.2% NAK
92% LEC + 8% NAK
84% LEC + 16% NAK

(a) 10% Loss Rate  (b) 15% Loss Rate  (c) 20% Loss Rate

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

Claim: Ricochet is reliable and time-critical.
Scalability in Groups
64 nodes, * groups per node, 10 nodes per group, Loss Model: Uniform 1%

Claim: Ricochet *scales* to hundreds of groups. Comparison: at 128 groups, SRM latency was 8 seconds. *400 times slower!*

Mahesh Balakrishnan  Ricochet: Lateral Error Correction for Time-Critical Multicast
Claim: Ricochet is *lightweight*

⇒ Time-Critical Apps can run over it
Motivation
System
Evaluation

Resilience to Burstiness
64 nodes, 128 groups per node, 10 nodes per group, Loss Model: Bursty 1%

... can handle short bursts (5-10 packets) well. Good enough?

Mahesh Balakrishnan
Ricochet: Lateral Error Correction for Time-Critical Multicast
Staggering
64 nodes, 128 groups per node, 10 nodes per group, Loss Model: Bursty 1%

Staggering of $i$: Encode every $i$th packet
Stagger 6, burst of 100 packets $\implies$ 90% recovered at 50 ms!
Multicast in Datacenters:
- large numbers of low-rate groups
- aggregate load can be high, causing packet loss

Ricochet is the first protocol to scale in the number of groups in the system

Layered under high-level platforms: Tempest, Axis2

Available for download:
<table>
<thead>
<tr>
<th>Motivation</th>
<th>System</th>
<th>Evaluation</th>
</tr>
</thead>
</table>

**Overflow**
Impact of Loss Rate on LEC
64 nodes, 128 groups per node, 10 nodes per group, Loss Model: *

Effect of Loss Rate on Recovery %

- Bursty b=10
- Uniform
- Markov m=10

Effect of Loss Rate on Latency

- Bursty b=10
- Uniform
- Markov m=10

Works well at typical datacenter loss rates: 1-5%