Reliable Group Communication

Presented by: Yuncheng Li
Outlines

• Reliable Multicast
  – Virtual Synchrony
    • Atomic Multicast

• Scalability
  – NACK
  – SRM (Feedback suppression)
  – Hierarchical control
  – Gossip protocol
Reliable Multicast

• Definition
  – All non-faulty members get the message

• Implication
  – Spread message
  – Handle message missing
  – Receive, buffer and deliver
Issues (non-implications)

• Reach membership agreement? \((Group\, view)\)
  – Two-general problem
  – Byzantine general problem

• Membership updates \((View\, changes)\)
  – Join or leave
  – Failed node detected

• Message order
Virtual Synchrony (view changes)

• Definition
  – Either delivered to all *(Group view)*
  – Or ignored by each of them

• Implication
  – Group view changes can abort multicast *(Barrier)*
An example

Figure 7-12. The principle of virtual synchronous multicast.
Message order

• Definition
  – Inter message constraints

• Types (Assume: virtual synchrony)
  – Unordered
  – FIFO
    • Receiver respects orders from each of the senders
  – Causality
    • Vector timestamp
  – Totally ordered
    • All receivers use the same order
FIFO message order

• Receiver respects orders of each senders
• Is the following FIFO order? Minimal correct?

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send m1</td>
<td>Recv m1</td>
<td>Recv m3</td>
<td>Send m3</td>
</tr>
<tr>
<td>Send m2</td>
<td>Recv m4</td>
<td>Recv m1</td>
<td>Send m4</td>
</tr>
<tr>
<td></td>
<td>Recv m2</td>
<td>Recv m2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recv m3</td>
<td>Recv m4</td>
<td></td>
</tr>
</tbody>
</table>
**Totally ordered Multicast**

- All receivers use the same order
- Is the following totally ordered? Minimal correction?

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send m1</td>
<td>Recv m1</td>
<td>Recv m1</td>
<td>Send m3</td>
</tr>
<tr>
<td>Send m2</td>
<td>Recv m4</td>
<td>Recv m4</td>
<td>Send m4</td>
</tr>
<tr>
<td></td>
<td>Recv m2</td>
<td>Recv m2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recv m3</td>
<td>Recv m3</td>
<td></td>
</tr>
</tbody>
</table>
Atomic Multicast

• Definition
  – Virtual Synchrony + Totally ordered

• All combinations

<table>
<thead>
<tr>
<th>Type of Multicast</th>
<th>Basic message ordering</th>
<th>Total-Ordered delivery?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable Multicast</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>FIFO Multicast</td>
<td>FIFO Ordered delivery</td>
<td>No</td>
</tr>
<tr>
<td>Causal Multicast</td>
<td>Causal Ordered delivery</td>
<td>No</td>
</tr>
<tr>
<td>Atomic Multicast</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>FIFO Atomic Multicast</td>
<td>FIFO Ordered delivery</td>
<td>Yes</td>
</tr>
<tr>
<td>Causal Atomic Multicast</td>
<td>Causal Ordered delivery</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Scalable Multicast

• Problem
  – How to spread message
  – How to deal with message missing

• Protocols
  – ACK / NACK
  – Feedback suppression
  – Hierarchical group control
  – Gossip protocol
ACK/NACK Feedback

Figure 7-8. A simple solution to reliable multicasting when all receivers are known and are assumed not to fail. (a) Message transmission. (b) Reporting feedback.
Feedback Suppression

Figure 7-9. Several receivers have scheduled a request for retransmission, but the first retransmission request leads to the suppression of others.
Hierarchical Group Control
Gossip Protocol
Gossip Protocol

• Algorithm,
  S1: Got a message
  S2: Random pick a node to send
  S3: If the node already know the message, stop spreading with a probability \((1/k)\)
  S4: Go back to S2
  S5: *Deliver the message* (assume all know)

• Theory:
  – \(s\): The fraction of nodes *ignorant*
  – \(k = 3\), \(s < 2\%\)
    \[ s = e^{-(k + 1)(1 - s)} \]
Gossip Protocol

• Analogy of epidemic or rumor spreading

Disclaimer: from last year
Other Update Propagation Model

• Anti-Entropy
  – Push Only
  – Pull Only
  – Push and Pull
  – Gossiping
    • Variable level of infectiveness – analogous to real life
    • Good propagation latency
    • No guarantee that all nodes will be eventually updated,
      \[ s = e^{\frac{1}{k+1}}(1-s) , \]
      \( k \) is the fraction of servers remain ignorant

Disclaimer: from last year
Summary

• ACK/NACK
  – Feedback flooding

• Feedback suppression
  – Hard to schedule NACK

• Hierarchical control
  – Hard to maintain topology

• Gossip protocol
  – Probabilistic, dynamic and approximation
  – Used by actual implementations
Questions?

- Reliable Multicast
- Virtual Synchrony
- Atomic Multicast

- ACK/NACK
- Feedback Suppression
- Hierarchical Control
- Gossip Protocol