Ashwin Lall

Fault-tolerant Group Communication

March 30th, 2005
Outline

- What is group communication?
- The Spinglass project
- Previous approaches to group communication
- Spinglass: The Epidemic model
- Spinglass Tools
- Applications
What is Fault-tolerant Group Communication?

• Why do we need reliable multi-casting schemes?

• What are some ways of ensuring reliable message delivery?

• Keeping track of group membership

• Application: Naval Defense
The Spinglass Project

• Focus on developing a scalable group communication scheme

• Uses an epidemic-style or gossip update mechanism

• Goal is to ensure scalability even when failures occur and during DOS attacks

• Previous schemes analyzed two cases: When every member worked, and when one went down. They did not study degradation in overall performance when individual members slowed down.
Two Flavors of Reliable Multicast

• **Virtual Synchrony** Model
  - Guarantees *fault-tolerance and consistency*
  - Automatically tracks *group membership*, reporting changes to members, and fault-tolerant multi-cast
  - Examples: Isis, Ensemble, Horus

• **Receiver-driven** Model
  - *Group membership* not tracked explicitly
  - Receivers are responsible for joining the group and requesting for data that they missed
  - Example: Scalable Reliable Multicasting (SRM)
Virtual Synchrony (e.g. Horus)

- Gives strong fault-tolerance guarantees

- If a process gets added to the group, or if a process fails, (called a view change) then want any message being passed either goes to every member of the group or to none of them.

- View changes act like a barrier to multicasts

- Each process maintains a stability flag for each message. If it detects a view change, it broadcasts all unstable messages before sending out the view change.
Disadvantages of Virtual Synchrony Model

- **Throughput Instability**

With even a small (10%) slowdown on a single member, the throughput of the entire system suffers noticeably.
Disadvantages of Virtual Synchrony Model

- **Micropartitions**

  Aggressive failure-detection is expensive since there is high overhead in removing a member from the group (micropartitioning), and possibly re-adding them later.

  Might drop healthy processes.

  Frequency of mistakes is at least linear and cost for each is linear, so costs are $O(n^2)$. 
Disadvantages of Virtual Synchrony Model

- **Convoys**

  To solve the above problem, the standard solution is to create a hierarchy of groups, organized in a tree.

  A hierarchical solution causes the data to be “bursty.”
Receiver-driven Models (e.g. SRM)

- Messages missed by some member are re-broadcast to all members.

- Each process that does not receive the message waits a randomly selected amount of time before requesting a re-broadcast.

- Goal is for only one process to issue the request for re-broadcast.

- Since there is a weaker reliability promise, should be more scalable.
Disadvantages of Receiver-driven Model

• Request and Retransmission Storms

Network latency may cause processes to miss each others requests before sending out their own. Found not to scale well:

Particularly susceptible to network flooding.
A Note on Message Ordering

1. Unordered multicast: No guarantees on order of message delivery

2. FIFO-ordered multi-cast: FIFO ordering for each member, but may vary across members

3. Causally-ordered multicast: Use vector timestamps

Totally-ordered multicast: Messages are delivered in the same order to all members

Atomic multicast: Total order AND virtual synchrony
Epidemic Protocols

- Modeled on the spread of an epidemic, or gossip, in a crowd

- Introduced at Xerox, developed further at Cornell

- Every process potentially communicates with any other process. If a process discovers an event, after $t$ time units, $O(2^t)$ processes have been informed of the event.
Epidemic Protocols - Example
Epidemic Protocols - Example
Epidemic Protocols - Example
Epidemic Protocols - Example
Bimodal Multicast

- Gossip-based protocol with two sub-protocols

- The first sub-protocol can be any unreliable data distribution protocol (e.g. IP multicast)

- The second sub-protocol is for recovery from missing messages. Each member maintains a random subset of the members of the group. At a set rate, processes choose a member from this set and send them a digest of its message buffer.

- If it is discovered that either the sender or the sendee is missing a message, then one process re-transmits that message to the other.
Bimodal Multicast - Analysis

\( R \): # of rounds of the protocol
\( N \): # of processes in the system
\( \beta N \): expected fanout of the gossip
\( \epsilon \): probability that a message between two non-faulty processes is lost (about 0.05)
\( \tau \): probability of process failure (about 0.001)

\( s_t \): # of processes that may gossip in round \( t \)
\( r_t \): # of processes that have not received gossip in round \( t \)
\( f_t \): # of processes that are faulty

\[
\begin{aligned}
  s_0 &= 1, r_0 = N - 1, f_0 = 0 \\
  r_{t+1} &= r_t - s_{t+1} \\
  \sum_{0 \leq t \leq R} s_t + r_R &= N
\end{aligned}
\]
Bimodal Multicast - Analysis

1. Probability of a process not getting a message, with $\epsilon$ fixed

$$p = \beta (1 - \epsilon), \quad q = 1 - p.$$  

$$Pr(X_i) = 1 - (1 - p)^{s_t} = 1 - q^{s_t}$$

2. Probability when an adversary can vary message-delivery probability between $[0, \epsilon]$  

$$\beta (1 - \epsilon) = p_{lo} \leq \beta (1 - \epsilon_{ij}) = p_{ij} \leq p_{hi} = \beta$$  

$$1 - q_{lo}^{s_t} = Pr(X_{lo}) \leq Pr(X_{j}) \leq Pr(X_{hi}) = 1 - q_{hi}^{s_t}$$

3. Now throw in faulty processes, $f_t$:  

$$1 - q_{lo}^{s_t-f_t} = Pr(X_{lo}) \leq Pr(X_{j}) \leq Pr(X_{hi}) = 1 - q_{hi}^{s_t}$$
Bimodal Multicast - Analysis

4. Lots of work...

\[
Pr(s_{t+1}|s_t, r_t, f_t) \leq \sum_{s_{t+1} \leq i \leq N} \binom{r_t}{i} (1 - q_{hi}^{s_t})^i (q_{hi}^{s_t})^{r_t-i} \\
- \sum_{s_{t+1} \leq i \leq N} \binom{r_t}{i} (1 - q_{lo}^{s_t-f_t})^i (q_{lo}^{s_t-f_t})^{r_t-i} \\
= R(s_t, r_t, f_t, s_{t+1})
\]

5. Using this value, the probability that the protocol ends in a failed state is (where \( \bar{f}_t = \sum_{0 \leq i \leq t} f_t \))

\[
F_t(s_t, r_t, \bar{f}_t) \leq \sum_{0 \leq f_t \leq s_t} \binom{s_t}{f_t} \tau^{f_t} (1 - \tau)^{s_t - f_t} \\
\max \sum_{0 \leq i \leq f_t} \sum_{0 \leq s_{t+1} \leq r_t} R(s_t, r_t, i, s_{t+1}) \\
F_{t+1}(s_{t+1}, r_t - s_{t+1}, \bar{f}_t + i)).
\]
Bimodal Multicast - Results from Analysis

Fig. 5. Number of susceptible processes versus number of gossip rounds when the initial multicast fails (left) and when it reaches 90% of processes (right; note scale). Both runs assume 1000 processes.
Bimodal Multicast - Results from Analysis

Scalability of P2cast reliability

\[ P(\text{failure}) \]

\begin{align*}
\text{#processes in system} & \quad 10 \quad 15 \quad 20 \quad 25 \quad 30 \quad 35 \quad 40 \quad 45 \quad 50 \quad 55 \quad 60 \\
1.0 \cdot 10^{-5} & \quad 1.0 \cdot 10^{-6} \quad 1.0 \cdot 10^{-7} \quad 1.0 \cdot 10^{-8} \quad 1.0 \cdot 10^{-9} \quad 1.0 \cdot 10^{-10} \quad 1.0 \cdot 10^{-11} \quad 1.0 \cdot 10^{-12} \quad 1.0 \cdot 10^{-13} \quad 1.0 \cdot 10^{-14} \quad 1.0 \cdot 10^{-15}
\end{align*}
Bimodal Multicast - Features

- **Constant load** on each of its members

- **Adaptable** reliability - trade-off between reliability and performance

- **Throughput stability**

- **Scalability (!!!)**
Security

• Processes should send signatures with data to avoid fake broadcasts

• Since message receipt is verified randomly across all members, there is a low probability that a process will miss a message due to flooding of the network

• No servers or single point of failure that can be targeted for an attack
Applications

- **Joint Battlespace InfoSphere**
  
  Maintains information for “publishers” and “subscribers” to uniformly access with focus on reliability (fast access) and allow rapid-change of the data.

- **Cluster Management**
  
  Lets each application believe that it has a dedicated platform.

- **Electric Power Grid**
  
  Allow *generators* in a region to monitor all others and vary their output based upon networked communication between them.
References


Thank you for your attention!