Primary-Backup Systems

CS249
FALL 2005
Sang Soo Kim
Active Replication vs. Primary-Backup

- **In active-replication (state machine approach from Ch.7)**
  - Client sends request to all servers
  - All servers execute all requests
  - Failure of any server is masked

- **In primary-backup systems**
  - Client sends request only to one designated primary server
  - Only the primary server executes the request
  - If the primary fails, one of the backup servers becomes the primary
A Primary-backup system is required to satisfy the following properties:

- **Pb1**: “There can be at most one primary at any time.”
- **Pb2**: “Client sends request to only to one server it thinks is the primary”
- **Pb3**: “Non-primaries do not process client requests.”
Formal Specification (Con't)

- Every request requires a response to be sent.
- A server outage occurs at time $t$ if a client sends a request at $t$ but receives no response.
- (k, $\Delta$)-bofo server: all server outages can be grouped into at most $k$ intervals of time, each having length of no more than $\Delta$.

- **Pb4**: “There exists fixed values $k$ and $\Delta$ such that the service behaves like a single (k, $\Delta$)-bofo server.”

![Diagram showing intervals $t_1$ and $t_2$ with (1, $t_2 - t_1$)-bofo]
A Simple Primary-Backup Protocol

- Primary sends **dummy messages** every $\tau$ seconds. If backup receives no such message for $\tau + \delta$ seconds, it becomes the primary.
- Service is $(1, \tau + 4\delta)$-bofo.
- The ordering of messages 2 & 3 matters!
  - (see next slide)
Suppose response is sent before state-update message.

- Then if the primary crashes before sending state-update message, the backup server is not properly updated.
- When the backup server becomes the primary later, the client will face inconsistency at the server.
Cost Metrics

1. **Degree of replication**: How many servers are needed to provide a $f$-fault-tolerant service?

2. **Blocking time**: What is the worst-case response time for a request?

3. **Failover time**: What is the worst-case interval during which requests can be lost because there is no primary?
Failure Models

- **Crash-failure**: A server may halt by crashing. It does not perform any further operations.

- **Crash + Link Failures**: A server may crash or a link may lose messages (but no delay, duplicate, or alter messages)

- **Send Omission**: A server fails by omitting to send some messages over non-faulty links or halting by crash.

- **Receive Omission**: A server fails by receiving only a subset of messages sent to it or by halting by crash.

- **General Omission Failures**: A server may exhibit send-omission or receive-omission failures.
## Lower Bounds: (1) Replication

<table>
<thead>
<tr>
<th>Failure Model</th>
<th>Degree of Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>$n &gt; f$</td>
</tr>
<tr>
<td>Crash + Link</td>
<td>$n &gt; f + 1$</td>
</tr>
<tr>
<td>Receive-Omission</td>
<td>$n &gt; \left\lfloor \frac{3f}{2} \right\rfloor$</td>
</tr>
<tr>
<td>Send-Omission</td>
<td>$n &gt; f$</td>
</tr>
<tr>
<td>General-Omission</td>
<td>$n &gt; 2f$</td>
</tr>
</tbody>
</table>

$n = \# \text{ of servers}$

$f = \# \text{ of failures that can be tolerated}$
Replication (con't)

- Crash and Send-omission: $n > f$
  - To tolerate $f$ failures, there should be at least $(f+1)$ servers in the system
Replication (con't)

- Crash + Link: $n > f + 1$
  - Proof by Contradiction

Diagram:
- Primary
- 1 Crashed!
- $f$ links
- $f$ links fail
- Primary in this group
- $A$
- $B$
Receive-Omission: $n > \lceil 3f / 2 \rceil$

Proof by Contradiction

Nodes in A and B commit receive-omission failures for messages sent from outside of their partitions.
Replication (con't)

- **General-Omission: \( n > 2f \)**
  - Proof by Contradiction

![Diagram](https://via.placeholder.com/150)

- Primary in this group
- \( f \) nodes commit send/receive omissions
# Lower Bounds: (2)Blocking Time

<table>
<thead>
<tr>
<th>Failure Model</th>
<th>Blocking Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>0</td>
</tr>
<tr>
<td>Crash + Link</td>
<td>0</td>
</tr>
<tr>
<td>Receive-Omission</td>
<td>$\delta$ when $f = 1$ and $n = 2$</td>
</tr>
<tr>
<td></td>
<td>$2\delta$ when $f &gt; 1$ and $n \leq 2f$</td>
</tr>
<tr>
<td></td>
<td>0 when $n &gt; 2f$</td>
</tr>
<tr>
<td>Send-Omission</td>
<td>$\delta$ when $f = 1$</td>
</tr>
<tr>
<td></td>
<td>$2\delta$ when $f &gt; 1$</td>
</tr>
<tr>
<td>General-Omission</td>
<td>$\delta$ when $f = 1$</td>
</tr>
<tr>
<td></td>
<td>$2\delta$ when $f &gt; 1$</td>
</tr>
</tbody>
</table>
Blocking Time (Con’t)

- Three Choices of blocking time:

1. **0 (Non-blocking):** primary can respond immediately to the client, knowing that state-update message is going to be eventually delivered

   - In Crash or crash+link failures, there is always one functioning path between the primary and the backups.
2. $\delta$: If $f=1$ (only one failure), we can have the backup server respond to the client. This adds a latency of $\delta$, due to the time that it takes for primary to send the update to the backup.
3. $2\delta$: If $f > 1$, have all the backups send ACKs to the primary, and have the primary respond to the client.
## Lower Bounds: (3) Failover Time

<table>
<thead>
<tr>
<th>Failure Model</th>
<th>Failover Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash</td>
<td>$f \delta$</td>
</tr>
<tr>
<td>Crash + Link</td>
<td>$2f \delta$</td>
</tr>
<tr>
<td>Receive-Omission</td>
<td>$2f \delta$</td>
</tr>
<tr>
<td>Send-Omission</td>
<td>$2f \delta$</td>
</tr>
<tr>
<td>General-Omission</td>
<td>$2f \delta$</td>
</tr>
</tbody>
</table>
Failover Time (Con't)

- Two Choices of Failover time:

1. \( f\delta \): For crash failures, the worst that can happen is the primary fails, followed by the next \( f-1 \) primaries failing immediately as they become primary. The message from the last primary, which actually survives, will be sent at time \( f\delta \) after the initial server died.

2. \( 2f\delta \): ???
Existing Primary-Backup Protocol: Alsberg-Day Protocol

- Primary and one backup
- If the request arrives at the primary:
  - Primary performs the requested operation
  - Sends update message to the backup and blocks
  - Backup updates its state, sends response to the client, and sends ACK to the primary.
Alsberg-Day Protocol (Con't)

- If the request arrives at a backup:
  - Backup forwards the request to the primary
  - Primary processes the request, sends response to the client, and sends a update message to the backup
  - Backup updates its state and discards the request

Diagram:

1. Request
2. Forward request
3. Response
4. Update

Primary -> Backup

Backup <-> Primary
Alsberg-Day Protocol (Con't)

- Failures are detected by lost ACK or by using "I-am-up" messages.

- If the backup crashes or the backup becomes primary, the primary recruits a new backup.
References

- *The Primary Backup Approach*. Navin Budhiraja, Keith Marzullo, Fred Schneider, Sam Toueg in “Distributed Systems” (Addison-Wesley)
- [www-users.itlabs.umn.edu/classes/ Spring-2002/csci8102/ClassNotes/Lecture7/Lecture7-6on1.pdf](www-users.itlabs.umn.edu/classes/ Spring-2002/csci8102/ClassNotes/Lecture7/Lecture7-6on1.pdf)