Review Paxos

Phase 1
- Prepare
  - Proposer: Send $\text{IAmLeader}(n)$ to all
  - Wait for a majority of responses
- Propose
  - Once majority is received, send $\text{Propose}(n,V)$ where $V$ is the highest-leader proposal among the responses (or my own value, if none of the responses had a value)

Phase 2
- Promise
  - If $n$ is the highest leader # I have seen: respond with $\text{YouAreLeader}(\text{Value}, \text{LeaderWhoProposedValue})$
- Accept
  - If $n$ is the highest leader # I have seen, send $\text{Accept}(n,V)$ to the learner

Reference: Manos Kapritsos, EECS 591 Distributed System, Lecture 10
Requirements in Paxos

- $2f + 1$ acceptors can tolerate $f$ failures
- In both phases, the leader must receive a majority of acceptors’ replies, which are Promises and Accepts respectively.
How Paxos works

Proposer A

Acceptor 1

Acceptor 2

Acceptor 3

Acceptor 4

Acceptor 5

Proposer B

Impossible to accept another value

Accept x
Flexible Paxos

• What will happen if a leader step into the next phase without a majority of replies?
• Why do we need a majority of Promises/Accepts in both phase?
• Do we really need a majority of responses in both phases?
How Paxos works

Proposer A

Acceptors 1, 2, 3

Proposer B
Flexible Paxos

- Quorum: a subset of participants (acceptors)
- Make sure at least one acceptor in common between phase 1 quorums ($Q_1$) and phase 2 quorums ($Q_2$).

Paxos: $|Q_1| > \frac{N}{2} \land |Q_2| > \frac{N}{2}$

Flexible Paxos: $|Q_1| + |Q_2| > N$
$|Q_1| = 4, |Q_2| = 2$
Simple Quorums

- Simple Quorums: Only consider the number of quorums in each phase

- Consider the case where $|Q2| < \frac{N}{2}$
  (Phase 2 is more common than phase 1 in practice)

- Tolerable failures become $|Q2| - 1$, but phase 2 can still be executed safely when failures $\leq N - |Q2|$
Trade-Offs

- Reduce the number of message from $4N$ to $2 \times (|Q_1| + |Q_2|)$
- By reducing the size of $Q_2$
  - Decrease latency and increase throughput
  - Require more acceptors to elect a new leader (reduce availability)
- May increase latency in some cases
Grid Quorums

- Ultimate goal: Find intersection between quorums in two phases
- Try rearranging the acceptors such that $N_1$ rows * $N_2$ cols = $N$
- No longer treat all failures equally

<table>
<thead>
<tr>
<th>Acceptors</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
Paxos: Grid Quorums

- Require a row and a column to form a quorum in both phases.

\[ \min(N_1, N_2) \leq f \leq (N_1 - 1) \times (N_2 - 1) \]

- Tolerate failures: \( \min(N_1, N_2) \leq f \leq (N_1 - 1) \times (N_2 - 1) \)
Flexible Paxos: Grid Quorums

- Require a row of acceptors for phase 1 and a column of acceptors for phase 2

- Tolerate failures: Depends on which set of acceptors failed
Flexible Paxos: Grid Quorums

- All failed nodes are within one column:
  - Paxos: Not tolerable
  - Flexible Paxos: Can continue execute phase 2 until new leader is needed
- All failed nodes are within one column:
  - Paxos: Not tolerable
  - Flexible Paxos: Can still try to elect a new leader and recover the process
Special cases

- $|Q_1| = N, |Q_2| = 1$
  - 1 acceptor is sufficient to form a $Q_2$
  - Unable to recover leader failure if any acceptor fails
- $|Q_1| = 1, |Q_2| = N$
  - All acceptors are required to form a $Q_2$
  - Any single acceptor is able to recover the leader failure
  => Can tolerate $f$ failures using $f + 1$ acceptors
Simulation

• Compare LibPaxos3 (Multi-Paxos) with FPaxos
• LibPaxos3: Send messages to all replicas
  FPaxos: Only send messages to a quorum of replicas
• Reducing $Q_2$ size => increase throughput and reduce latency

**Figure 4** Throughput and average latency of FPaxos with various quorum sizes and LibPaxos3.
Summary

- Flexible Paxos weakens the majority quorum constraint in Paxos.
- As long as $Q_1$ and $Q_2$ are guaranteed to intersect, it can form a Flexible Paxos.
- Reduce the latency and increase the throughput.
- Generally decrease the number of tolerate failures, but one phase can still work in some situations.
Q & A