Reconfigurable Replicated State Machine

Seif Haridi
KTH
Motivation

- A Replicated State Machine (RSM) is running on a set of N processes (typically 3 or 5)
- Can tolerate up to \( \lceil N/2 \rceil \) process failures
  - One more and RSM unavailable
  - Need a way to replace faulty processes
- Impossible to know if a process is faulty or slow in asynchronous system
  - Must be able to replace any process,
  - This is called reconfiguration
Policy (when) vs Mechanism (how)

- External agent decides when to reconfigure
  - Autonomous management system
  - Or human operator
- The agent chooses new configuration
  - Example: $\Pi_{\text{old}} = \{p_1, p_2, p_3\} \Rightarrow \Pi_{\text{new}} = \{p_1, p_2, p_4\}$
    - Can, in general, be a completely new set of processes
      - $\Pi_{\text{old}} \cap \Pi_{\text{new}} = \emptyset$
    - Often a single suspected process is replaced
- Only concerned with mechanism
  - Leave the policy as open and flexible as possible
Configurations

- Each configuration is conceptually an instance of Sequence-Paxos

- Replicas in configuration $c_0 = \{r_{01}, r_{02}, r_{03}\}$
- A process $p_1$ may act as multiple replicas
  - In different configurations, for example $\{r_{01}, r_{11}, r_{21}\}$
Configurations

- Each configuration is conceptually an instance of Sequence-Paxos

- Replicas in configuration $c_1 = \{r_{11}, r_{12}, r_{13}, r_{14}\}$

- A process may act as multiple replicas in different configurations,
  - for example, $p_1$ is $\{r_{01}, r_{11}, r_{21}\}$
Configurations

- RSM executes in a configuration until a reconfiguration occurs, then moves to new configuration
  - Processes that move to the new configuration from the previous one does that asynchronously
  - Once a majority of processes have moved/entered the new configuration is active
- The first configuration, $c_0$, starts with the empty sequence accepted in round 0
- It then runs normally
  - If sequence $v$ is issued in round $n$ then $v$ is an extension of all sequences chosen in rounds $\leq n$
Ballot Array of \( c_0 \)

- Replicas \( r_{0,1}, r_{0,2} \) and \( r_{0,3} \) in configuration \( c_0 \)

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by ( r_{0,1} )</th>
<th>Accepted by ( r_{0,2} )</th>
<th>Accepted by ( r_{0,3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ldots )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n=2 )</td>
<td>( \langle C_2 \rangle )</td>
<td>( \langle C_2 \rangle )</td>
<td></td>
</tr>
<tr>
<td>( n=1 )</td>
<td>( \langle C_1 \rangle )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n=0 )</td>
<td>( \langle \rangle )</td>
<td>( \langle \rangle )</td>
<td>( \langle \rangle )</td>
</tr>
</tbody>
</table>

- Empty sequence accepted in round 0
- If sequence \( v \) is issued in round \( n \) then \( v \) is an extension of all sequences chosen in rounds \( \leq n \)
Stop-sign in $c_0$

- At some point, a special *stop-sign* command is proposed, and a proposer extends the current sequence with this command.
- The sequence with the *stop-sign* as last command is the final sequence in $c_0$.
- No sequence longer than the final sequence in $c_0$ may be issued by any proposer in $c_0$.
- Therefore, after the final sequence is chosen, no longer sequence can be chosen in $c_0$, and $c_0$ is *stopped*.
Final Sequence in $c_0$

- Replicas $r_{0,1}$, $r_{0,2}$ and $r_{0,3}$ in configuration $c_0$

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{0,1}$</th>
<th>Accepted by $r_{0,2}$</th>
<th>Accepted by $r_{0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=3</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=2</td>
<td>$\langle C_2 \rangle$</td>
<td>$\langle C_2 \rangle$</td>
<td></td>
</tr>
<tr>
<td>n=1</td>
<td>$\langle C_1 \rangle$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=0</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
</tr>
</tbody>
</table>

- $SS_0$ is the stop-sign command in $c_0$
- The final sequence in $c_0$ is $\sigma_0 = \langle C_2, SS_0 \rangle$
- Any Sequence in rounds $n > 3$ will be $\sigma_0$
Final Sequence in $c_0$

- Replica $r_{0,1}$ and $p_1$ crashed at round 3 after $\sigma_0$ is chosen, $r_{0,2}$ or $r_{0,3}$ proposes

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{0,1}$ at $p_1$</th>
<th>Accepted by $r_{0,2}$ at $p_2$</th>
<th>Accepted by $r_{0,3}$ at $p_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=3</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td></td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=2</td>
<td></td>
<td>$\langle C_2 \rangle$</td>
<td>$\langle C_2 \rangle$</td>
</tr>
<tr>
<td>n=1</td>
<td>$\langle C_1 \rangle$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=0</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
</tr>
</tbody>
</table>

- The final sequence in $c_0$ is $\sigma_0 = \langle C_2, SS_0 \rangle$
- Eventually all correct replicas has decided on $SS_0$
Starting New Configurations
Starting a New Configuration

- Once the final sequence \((v_a)\) is decided \(\sigma_i\) in \(c_i\) and is in the persistent store by one process \(p\) the new configuration \(c_{i+1}\) can start
  - \(C_i\) is stopped
- \(SS_i\) has complete information about the \(c_{i+1}\)
  - \(\Pi_{i+1}:\) the set of processes in \(c_{i+1}\)
  - \(cfg:\) new configuration number
  - \(RID:\) for each process \(p_j\) in \(\Pi_{i+1}\) its replica identifier \(r_{i+1,j}\)
- Each correct \(p_j\) that is in both in \(c_i\) and \(c_{i+1}\) waits for its replica \(r_{i,j}\) to decide its final sequence \(\sigma_i\) before taking it over as its initial sequence
- Each correct \(p_j\) not in \(c_i\) copies its initial sequence \(\sigma_i\) from persistent store
Starting a New Configuration

- Each process $p_j$ starts its new replica in $c_{i+1}$
- For each replica, the final sequence in $c_i$ is automatically accepted in round 0 of $c_{i+1}$
  - Round 0 in each configuration is special
  - Other rounds work as normal
- In leader-based consensus each configuration has its own leader election abstraction
- the leader election abstraction starts when the new configuration starts
Initial Sequence in $c_1$

- Replicas $r_{1,1}$, $r_{1,2}$ and $r_{1,4}$ in configuration $c_1$

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{1,1}$ at $p_1$</th>
<th>Accepted by $r_{1,2}$ at $p_2$</th>
<th>Accepted by $r_{1,4}$ at $p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=0$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
</tbody>
</table>

- The final sequence in $c_0$ is chosen in round 0 in configuration $c_1$
- $P_3$ is removed
Execution in $c_1$

- Replicas $r_{1,1}$, $r_{1,2}$ and $r_{1,4}$ in configuration $c_1$

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{1,1}$ at $p_1$</th>
<th>Accepted by $r_{1,2}$ at $p_2$</th>
<th>Accepted by $r_{1,4}$ at $p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=3</td>
<td>$\langle C_2, SS_0, C_3, C_5 \rangle$</td>
<td>$\langle C_2, SS_0, C_3, C_5 \rangle$</td>
<td></td>
</tr>
<tr>
<td>n=2</td>
<td></td>
<td></td>
<td>$\langle C_2, SS_0, C_3, C_4 \rangle$</td>
</tr>
<tr>
<td>n=1</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
</tr>
<tr>
<td>n=0</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
</tbody>
</table>

- The final sequence in $c_0$ is chosen in round 0 in configuration $c_1$
Overlapping configurations

\[ C_{\text{old}} \]

\[ C_{\text{new}} \]

\( p_1 \)

\( p_2 \)

\( p_3 \)

\( p_4 \)

\( p_5 \)

S. Haridi, KTHx ID2203.2x
Extend Round Numbers

- Round numbers are extended to pairs \((c, n)\)
  - \(c\) is a configuration and \(n\) is local round number within that configuration
- Since configurations are totally ordered, rounds are totally ordered across configurations
### Ordering Rounds Totally

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{c1,1}$</th>
<th>Accepted by $r_{c1,2}$</th>
<th>Accepted by $r_{c1,4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=(c₁, 3)</td>
<td>$\langle C_2, SS_0, C_3, C_5 \rangle$</td>
<td>$\langle C_2, SS_0, C_3, C_5 \rangle$</td>
<td></td>
</tr>
<tr>
<td>n=(c₁, 2)</td>
<td></td>
<td></td>
<td>$\langle C_2, SS_0, C_3, C_4 \rangle$</td>
</tr>
<tr>
<td>n=(c₁, 1)</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
<td>$\langle C_2, SS_0, C_3 \rangle$</td>
</tr>
<tr>
<td>n=(c₁, 0)</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round</th>
<th>Accepted by $r_{c0,1}$</th>
<th>Accepted by $r_{c0,2}$</th>
<th>Accepted by $r_{c0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=(c₀, 3)</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
<td>$\langle C_2, SS_0 \rangle$</td>
</tr>
<tr>
<td>n=(c₀, 2)</td>
<td></td>
<td>$\langle C_2 \rangle$</td>
<td>$\langle C_2 \rangle$</td>
</tr>
<tr>
<td>n=(c₀, 1)</td>
<td>$\langle C_1 \rangle$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=(c₀, 0)</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
<td>$\langle \rangle$</td>
</tr>
</tbody>
</table>
Starting/Stopping Configuration

- A process has replicas in multiple configurations
  - But can only be **running** in **one** configuration at any time
- Starting and stopping configuration $c_i$ is not coordinated between processes
  - Process $p_i$ may have a stopped replica at $c_{i-1}$ helping other replicas to reach the final sequence in $c_{i-1}$
  - Process $p_i$ may have a replica in $c_{i+1}$ that did not start yet
Configurations

- Each configuration is conceptually an instance of Sequence-Paxos

- A process $p$ may act as multiple replicas
  - in different configurations, for example $\{r_{11}, r_{21}, r_{31}\}$
  - $p$ is stopped in $c_1$, running in $c_2$, not-started in $c_3$
Hand-Over

- As soon as proposer p in $c_i$ learns that the final sequence $\sigma_i$ in $c_i$ is chosen, p should inform replicas in $c_{i+1}$ about $\sigma_i$
  - So that replicas in $c_{i+1}$ can start, to maintain availability
- Process that acts as replica in both $c_i$ and $c_{i+1}$ will learn $\sigma_i$ through normal Decide msgs
- Other replicas in $c_{i+1}$ must also learn $\sigma_i$
  - Send additional Decide messages
Optimizations
Hand-Over: Early Sequence Transfer

- As leader $p$ receives proposal with stop-sign command SS, one possibility is for $p$ to not issue proposal with SS immediately.
- Instead, $p$ starts to update replicas in $c_{i+1}$ with longest sequence decided so far.
- Only sends accepts SS once replicas in $c_{i+1}$ are sufficiently up to date.
- This way the interruption in service is minimized.
  - Trade-offs
Efficient Hand-Over

- If new replica process \( p \) is introduced in system then entire sequence has to be transferred to \( p \)
  - This may take some time
- Dividing the sequence into chunks, and letting other processes in the same configuration send chunks in parallel to \( p \) may increase efficiency
  - Trade-off between on network capacity, processor load, etc.
Snapshots

- Currently, the entire sequence of commands must be transferred from a configuration to the next specially when new processes are introduced.
- It is possible to take a snapshot of the state of the state machine after it has executed a certain number of commands:
  - E.g. every 1000 commands or every 1h.
  - Can then garbage collect prefix of sequence.
- Care must be taken to still prevent duplicate commands?
Prepare Phase

- A leader $p$ in $c_i$ sends Prepare messages to all replicas in $c_i$

- For each process $q$, there are three cases:
  1. $q$’s replica is running in $c_i$: $q$ behaves normally
  2. $q$’s replica hasn’t started in $c_i$ yet: $q$ obtains final sequence in $c_{i-1}$ from $p$, starts its replica in $c_i$, and sends Promise to $p$
  3. $q$’s replica has stopped $c_i$: $q$ sends the final sequence in $c_i$ to $p$, after which $p$ will also stop and starts its replica in $c_{i+1}$

- In case 1 and 2, $p$ waits for Promise messages from a majority
Summary

- Reconfiguring a replicated state machine is relatively straightforward.
- Round numbers are extended so that rounds in an earlier configuration are ordered before rounds in a later configuration.
- At most one of the replicas that a process implements may be running at any time.
- The hand-over procedure is important in order to get availability and efficiency.