Understanding Self-healing in Service Discovery Systems

Chris Dabrowski and Kevin Mills

WOSS 2002
Charleston, South Carolina
November 18, 2002
Observations on Self-healing in Distributed Systems

• Recovery strategies are critical for self-healing as failure rate increases.
  – More so than other factors (e.g., architecture, topology, consistency-maintenance mechanisms)

• Recovery strategies can interact in complex and unexpected ways
  – Redundancy (only one is necessary)
  – Complimentaryness (both are necessary)
  – Interference (one strategy prevents another from succeeding)

• When designing self-healing distributed systems based on service discovery protocols, need to consider:
  – The types of failure expected and their likelihood
  – Detailed protocol behaviors (e.g., discovery, update propagation, recovery) and not simply the application-programming interface.
Dynamic discovery protocols in essence... enable *distributed software components*

1. To **discover** each other without prior arrangement,
2. To **express** opportunities for collaboration,
3. To **compose** themselves into larger collections that cooperate to meet an application need, and
4. To **detect and adapt** to failures.

**Some examples:**

<table>
<thead>
<tr>
<th>3-Party Design</th>
<th>2-Party Design</th>
<th>Adaptive 2/3-Party Design</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JINI</strong></td>
<td><strong>UPnP</strong></td>
<td><strong>SLP</strong></td>
</tr>
<tr>
<td>Vertically Integrated 3-Party Design</td>
<td>Network-Dependent 3-Party Design</td>
<td>Network-Dependent 2-Party Design</td>
</tr>
</tbody>
</table>
Self-healing in Hostile and Volatile Conditions

Service discovery systems must ensure consistency of information about services in failure environments.

Contributing factors: recovery strategies, architectures, topologies, and consistency-maintenance mechanisms (polling & notification).

This study focuses on role of recovery strategies.
Two Generic Architectures Underlie Six Discovery Protocols

**Update Propagation Method**

§ **Notification** – Updates forwarded by Managers immediately after they occur.

Service Users request leases with Service Managers to obtain notifications. Notifications rely on TCP for robustness, but TCP may fail and issue a remote exception.
Understanding Contribution of Failure Detection and Recovery Strategies to Update Effectiveness

Types of Strategies:

- Application Persistence
  Application-specific behaviors, including responses to remote exceptions: (1) ignore, (2) bounded retries, and/or (3) discard local knowledge of remote components

- Remote Exceptions

- TCP attempts reliable delivery

- Unicast UDP no delivery guarantees

- Multicast UDP no delivery guarantees

- Soft State
  (Re) discovery of services after purge after lost periodic announcements

- The Network

- API Boundary
  Application persistence behaviors required by the discovery protocol
Consistency Maintenance Using Notification

For All (SM, SU, SD):

\[(SM, SD \text{ [Attributes1]} ) \text{ IsElementOf SU discovered-services}
\]
\[SD \text{ [Attributes2]} \text{ IsElementOf SM managed-services}
\]
implies Attributes1 = Attributes2

Scenario

How well does the system restore consistency after failure?

TCP retries fail; remote exception ignored
Soft State Recovery of Service After Failed Notification

Scenario

SM

Discovered-Services

Announcement (SD)

Get Description(SD)

Description Response (SD)

Notification Request(SD)

Notification Request Accepted

Managed-Services (SD[Attribute1])

Update (SD[Attribute2])

Notification (SD[Attribute2])

Managed-Services (SD[Attribute2])

SU

Consistency Condition Violated

Time Out and SU Purge

TCP retries fail & remote exception ignored, BUT recovery occurs through soft state

Search Query

Query Response (SD)

Get Description(SD)

Description Response (SD)

Consistency Restored!
Application Persistence Recovery of Service After Failed Notification

Scenario

SM

Managed-Services (SD[Attribute1])

Update (SD[Attribute2])

Managed-Services (SD[Attribute2])

SU

Consistency Condition Violated

Consistency Restored!

TCP retries fail, BUT recovery occurs through application persistence in response to remote exception
1. Choose a time to introduce the change [uniform(Q, D/2)]
2. For each node, choose a time to introduce an interface failure [uniform(Q, D-(D*F))]
3. When each interface failure occurs, choose the scope of the failure, where each of [Rx, Tx, Both] has an equal probability

Q = end of quiescent period (100 s in our experiment)
D = propagation deadline (5400 s in our experiment)
F = Interface Failure Rate (variable from 0% - 75% in 5% increments in our experiment)
Modeling and Analysis Approach

**Scenario**

**Modeling and Analysis Approach**

**Aggressive Discovery Multicast Group**

**Parameters**

<table>
<thead>
<tr>
<th>Time</th>
<th>Command</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>NodeFail</td>
<td>SM4</td>
</tr>
<tr>
<td>5</td>
<td>LinkFail</td>
<td>SCM1 SM4</td>
</tr>
<tr>
<td>10</td>
<td>GroupJoin</td>
<td>SM4 GROUP1</td>
</tr>
<tr>
<td>10</td>
<td>FindService</td>
<td>SU8 5 1 2 S XYZ ALL</td>
</tr>
<tr>
<td>50</td>
<td>AddService</td>
<td>SM4 SCM3 T ATT API GUI 20 30</td>
</tr>
</tbody>
</table>

**Execute with Rapide**

**Analyze POSETs**

Use metrics to Assess Correctness & Performance
Update Effectiveness in Response to Interface Failure

<table>
<thead>
<tr>
<th></th>
<th>Both Recovery Strategies</th>
<th>Soft State Only</th>
<th>Application Persistence Only</th>
<th>No Recovery Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Party Notification</td>
<td>0.915</td>
<td>0.853</td>
<td>0.836</td>
<td>0.431</td>
</tr>
<tr>
<td>Three-Party Notification Single SCM</td>
<td>0.819</td>
<td>0.816</td>
<td>0.828</td>
<td>0.383</td>
</tr>
<tr>
<td>Three-Party Notification Dual SCM</td>
<td>0.856</td>
<td>0.879</td>
<td>0.887</td>
<td>0.465</td>
</tr>
</tbody>
</table>

Two-party

Three-party Single SCM

Three-party Dual SCM
Results

- **Under Conditions of Interface Failure**
  - Performance decreases linearly in absence of recovery strategies
  - Soft State alone:
    - In both architectures, discovery discard decreases time available to recover.
    - In two-party, Soft State recovery alone is insufficient because recovery is not stimulated when failures block Get Description Requests or Notifications, but not announcements.
    - In three-party, Soft State alone approaches performance of both strategies together, because discovery discarded after same period as when both strategies used together.
  - Application Persistence alone:
    - In two-party, Application persistence may be sufficient, but in our experiments it’s limited by lease renewal algorithm (residual 2.5% not renewed).
    - In three-party, Application Persistence performs as well as both strategies together because retries continue every 120s.
    - If additional SCMs provided, more paths for recovery and propagation allow Application Persistence to exceed both strategies together.
Message Loss Model for Experiment

1. Choose a time to introduce the change \([\text{uniform}(Q, D/2)]\)
2. For each message transmission, determine if message is lost using \(F\)

Q = end of quiescent period (100 s in our experiment)
D = propagation deadline (5400 s in our experiment)
F = message loss rate (variable from 0% - 95% in 5% increments in our experiment)
Update Effectiveness in Response to Message Loss

<table>
<thead>
<tr>
<th></th>
<th>Both Recovery Strategies</th>
<th>Soft State Only</th>
<th>Application Persistence Only</th>
<th>No Recovery Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Party Notification</td>
<td>0.914</td>
<td>0.715</td>
<td>0.921</td>
<td>0.675</td>
</tr>
<tr>
<td>Three-Party Notification Single SCM</td>
<td>0.913</td>
<td>0.781</td>
<td>0.954</td>
<td>0.679</td>
</tr>
<tr>
<td>Three-Party Notification Dual SCM</td>
<td>0.964</td>
<td>0.877</td>
<td>0.994</td>
<td>0.787</td>
</tr>
</tbody>
</table>

Two-party

Three-party Single SCM

Three-party Dual SCM
Results

- **Under Conditions of Message Loss**
  - Again, performance decreases linearly without recovery strategy
  - In three-party architecture, additional SCMs provide more paths for propagation and recovery.
  - Soft State alone:
    - Performance under Soft State alone insufficient because after discovery discard, rediscovery messages continue to be subject to message loss (making it harder to rediscover at high failure rates).
  - In Application Persistence alone
    - Application Persistence better than both strategies together because retries continue every 120s AND additional messages for rediscovery are not used.

- However, if nodes fail and are replaced by new nodes (different experiment), Soft State becomes more important than Application Persistence.
Observations on Self-healing in Distributed Systems

- Recovery strategies are critical for self-healing as failure rate increases.
  - More so than other factors (e.g., architecture, topology, consistency-maintenance mechanisms)

- Recovery strategies can interact in complex and unexpected ways
  - Redundancy (only one is necessary)
  - Complimentaryness (both are necessary)
  - Interference (one strategy prevents another from succeeding)

- When designing self-healing distributed systems based on service discovery protocols, need to consider:
  - The types of failure expected and their likelihood
  - Detailed protocol behaviors (e.g., discovery, update propagation, recovery) and not simply the application-programming interface.
Extra
Update Effectiveness of Two-Party Notification
(For Interface Failure with Soft State & Application Persistence Strategies Factored)
Update Effectiveness of Three-Party Notification, Single SCM (For Interface Failure, with Soft State & Application Persistence Factored)
Update Effectiveness of Three-Party Notification, Dual SCM
(For Interface Failure, with Soft State & Application Persistence Factored)
Update Effectiveness of Two-Party Notification (For Message Loss, with Soft State & Application Persistence Factored)
Update Effectiveness of Three-Party Notification, Single SCM (For Message Loss, with Soft State & Application Persistence Factored)
Update Effectiveness of Three-Party Notification, Dual SCM
(For Message Loss, with Soft State & Application Persistence Factored)

- Both Recovery Strategies
- Soft State Only
- Application Persistence Only
- No Recovery Strategy

Effectiveness vs. Failure Rate (%)