Self-Adaptive Discovery Mechanisms
for Improved Performance in Fault-Tolerant Networks

Kevin Mills, Chris Dabrowski, and Jesse Elder

DARPA FTN PI Meeting
January 18, 2002

Survivable Software for Harsh Environments
Presentation Outline

- One-Page Review of Project Objective and Plan
- Brief Refresher on Service Discovery Protocols
- Outline of Technical Approach to Understand Fault-Tolerant Behavior of Service Discovery Protocols
- Initial Results Comparing Behavior of Jini and Universal Plug-and-Play when Propagating Information during Interface Failures
- Plan for Next Six Months
- Conclusions

More Details Available on Supplementary Slides
Project Objective

Research, design, evaluate, and implement self-adaptive mechanisms and algorithms to improve the performance of service discovery protocols for use in fault-tolerant networks.

Project Plan – Three Phases

• **Phase I** – characterize performance of selected service discovery protocols (Universal Plug-and-Play – UPnP – and Jini) as specified and implemented
  - develop simulation models for each protocol
  - establish performance benchmarks based on default or recommended parameter values and on required or most likely implementation of behaviors

• **Phase II** – design, simulate, and evaluate self-adaptive algorithms to improve performance of discovery protocols regarding selected mechanisms
  - devise algorithms to adjust control parameters and behavior in each protocol
  - simulate performance of each algorithm against benchmark performance
  - select most promising algorithms for further development

• **Phase III** – implement and validate the most promising algorithms in publicly available reference software
Dynamic Discovery Protocols in Essence

Dynamic discovery protocols enable network elements (including software clients and services, as well as devices):
(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 changes in network topology.

Selected First-Generation Dynamic Discovery Protocols

<table>
<thead>
<tr>
<th>3-Party Design</th>
<th>Universal Plug and Play Design</th>
<th>Adaptive 2/3-Party Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jini</td>
<td>SLP for Enterprise Networks</td>
<td></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>
Two Party vs. Three Party Architectures

**Third Party**

- Service Information Collection

**SERVICE MANAGER**
- Discover Network Context()
- <<opt>> Announce Service Processing()
- Service Manager()
- <<not shr>> start Service Parameter Matching Task()

**SERVICE CACHE MANAGER**
- Service Repository
- Service Description
- Service Provider
- Service User
- LOCAL CACHE MANAGER
- 0..*
- Service Information Collection
- Service Availability Requests
- Notification Request
- Notification Cache
- Notification Request
- Service Availability Requests
- Service Repository
- Service Description
- Service Provider
- Service User
- LOCAL CACHE MANAGER
- 0..*

**SERVICE PROVIDER**
- Parameter Notification Request
- Service Repository
- Service Description
- Service User
- LOCAL CACHE MANAGER
- 0..*

**SERVICE USER**
- Parameter Notification Request
- Service Repository
- Service Description
- Service Provider
- LOCAL CACHE MANAGER
- 0..*

**LOCAL CACHE MANAGER**
- Start Aging Task()

**1/18/2002**
Technical Approach to Phase I – Use Rapide to Model and Understand Fault Behavior of Jini and UPnP

### Specification Model

#### **3.3 DIRECTED DISCOVERY CLIENT INTERFACE**

- **This is used by all Jini entities in directed discovery mode.** It is part of the SCM_Discovery Module. Sends Unicast messages to SCMs on list of SCMS to be discovered until all SCMS are found.
- Receives updates from SCM DB of discovered SCMs and removes SCMs accordingly.
- NOTE: Failure and recovery behavior are not yet defined and need review.

**TYPE Directed_Discovery_Client**

(SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code; InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)

**IS INTERFACE**

**SERVICE DDC_SEND_DIR** : DIRECTED_2_STEP_PROTOCOL;

**SERVICE DISC_MODES** : dual SCM_DISCOVERY_MODES;

**SERVICE DD_SC_INCREMENT** : DD_SC_INCREMENT;

**SERVICE SCM_Update** : SCM_Update;

**SERVICE DB_Update** : dual DB_Update;

**SERVICE NODE_FAILURES** : NODE_FAILURES; -- events for failure and recovery.

**ACTION**

IN Send_Requests(),
BeginDirectedDiscovery();

**BEHAVIOR**

action animation_Iam (name: string);

MySourceID : VAR IP_Address;

PV : VAR ProtocolVersion;

1/18/2002
Two-Party (UPnP) Topology for Experiment

UPnP Multicast Group

Service User

HTTP/TCP Unicast Links

HTTP/UDP Unicast Links

Service Manager

Five SUs

One SM
Three-Party (Jini) Topologies for Experiment

- Service Manager
- Service User
- Service Cache Manager

- Aggressive Discovery
- Remote Method Invocation
- Unicast Links
- Lazy Discovery

- One SM
- Five SUs
- Optional 2nd SCM
- One SCM
Faults Interfere with Discovery and Information Propagation

- **Interface Failure – Tx, Rx, or Both**
  - Due to nearby enemy jamming or other interference
  - Due to multi-path fading during mobility

- **Path Loss – Pt-Pt or Area-Area and Full-duplex or Half-duplex**
  - Due to persistent congestion
  - Due to physical link cuts
  - Due to enemy jamming at routers

- **Message Loss – under both UDP and TCP (>delay)**
  - Due to sporadic or distant enemy jamming or other interference
  - Due to transient congestion
  - Due to multi-path fading during mobility

- **Node Failure – Partial or Complete with variable persistence of information**
  - Due to enemy bombardment or cyber attacks
  - Due to mobility associated with military operations
**Discovery Systems Divide Recovery Responsibilities: Lower Layers, Discovery Protocol, and Application**

- **Selective Reliable Delivery by Lower Layers**
  - TCP attempts retransmissions (basis for Jini-RMI and UPnP-HTTP)
  - UDP messages in UPnP sent as multiple copies

- **Periodic Announcements by Discovery Protocol**
  - Allows caching nodes to discard information when TTL expires
  - Jini includes aggressive search at node start up, while UPnP permits nodes to undertake aggressive search at any time

- **Periodic Refreshing of Resources Required by Discovery Protocol**
  - Allows resource owner to free resource when refresh period expires

- **Remote Exceptions Issued by Protocol Over TCP Links**
  - Allows application to take recovery action: Ignore? Retry? Discard knowledge of service or resource?
Understanding Information Propagation in Discovery Architectures during Interface Failure

How do various service discovery architectures, topologies, and fault-recovery mechanisms perform under deadline during interface failure?

Outline of Experiment

- Deploy models of two-party (UPnP) and three-party (Jini) architectures with one SM and five SUs (for Jini include two topologies – one and two SCMs).

- Ensure initial discovery and information propagation completed.

- Introduce a change in the service description at the SM, and establish a deadline for propagating the new information to all SUs.

- Measure the number of messages exchanged and the latency required to propagate the information to all SUs, or until the deadline arrives, under two different propagation mechanisms: polling and eventing.

- Repeat this experiment while varying the percentage of interface failure time for each node up to 75% (in increments of 5%).
Information-Propagation Mechanisms for Experiment

Polling

Two Party

- Change Service
- Get Description
- Get Description
- Get Description

Three Party

- Change Service
- Get Description
- Get Description
- Get Description

Eventing

- SM
- SU
- Notification Request
- Get Description
- Notification
- Get Description

- SM
- SU
- Notification Request
- Find Service
- Notification
- Find Service
- Change Service

1/18/2002
Interface-Failure Model for Experiment

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)

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-Q)*F))]
3. When each interface failure occurs, choose the scope of the failure, where each of [Rx, Tx, Both] has an equal probability

Change introduced sometime in this interval
Interface failures occur sometime during this interval
Interface failures repaired after appropriate time

Random Processes

TIME

Discovery occurs & initial information propagated

1/18/2002 13
Our Model Responses to Remote Exceptions (“Approximately”)

- Ignore REX Received
  - When replying to a remote-method invocation
  - When attempting to cancel a lease
  - When attempting to renew a lease (But then attempt to obtain a new lease)

- Retry Operation for Some Period of Time - Then Quit If Not Successful
  (If Quitting, Eliminate Local Knowledge of Discovered Entity)
  - When attempting to register for notification events
  - When a UPnP SU requests service descriptions

- Retry Operation Persistently as Long as Application Executes
  - When a Jini SM attempts to register a service description with a SCM
Metrics Devised for Information-Propagation Experiments

- **Update Responsiveness** \((R)\)
  Assuming that information is created at a particular time and must be propagated by a deadline, then the difference between the deadline and the creation time represents available time in which to propagate the information. Update Responsiveness, \(R\), measures the *proportion of the available time remaining after the information is propagated*. \([1 = \text{all time remains} \text{ and } 0 = \text{no time remains}]\)

- **Update Effectiveness** \((U)\)
  Update Effectiveness, \(U\), measures the *probability that information will propagate successfully to a SU before some deadline, \(D\)*. \([1 = \text{information will be propagated} \text{ and } 0 = \text{information will not be propagated}]\)

- **Update Efficiency** \((E)\)
  Given a specific topology of SUs and SMs in a discovery system, examination of the available architectures (two-party and three-party) and mechanisms (polling and eventing) will reveal a minimum number of messages that need to be sent to propagate information from all SMs to all SUs in the topology. Update Efficiency, \(E\), can be measured as the *ratio of the minimum number of messages needed to the actual number of messages observed*. \([1 = \text{only minimum number of messages needed} \text{ and } 0 = \text{infinite number of messages needed}]\)
Responsiveness: UPnP (2-Party) vs. Jini (3-Party)
Effectiveness: UPnP (2-Party) vs. Jini (3-Party)

Update Effectiveness vs. Interface Failure Rate (%)

- 2-Party Notification
- 2-Party Polling
- 3-Party Notification 1 SCM
- 3-Party Polling 1 SCM
- 3-Party Notification 2 SCMs
- 3-Party Polling 2 SCMs
Efficiency: UPnP (2-Party) vs. Jini (3-Party)

- 2-Party Notification
- 2-Party Polling
- 3-Party Notification 1 SCM
- 3-Party Polling 1 SCM
- 3-Party Notification 2 SCMs
- 3-Party Polling 2 SCMs

Average Update Efficiency vs. Interface Failure Rate (%)

Graph showing efficiency comparison between UPnP (2-Party) and Jini (3-Party) systems.
Plan for the Next Six Months

- Submit two papers on recent results: MILCOM 2002 and 3rd International Workshop on Software Performance

- Complete characterization of UPnP and Jini behavior (*ending Phase I*)
  - Information propagation during message loss
  - State recovery during node failure
  - Performance under increasing network size

- Develop and document ideas for initial set of self-adaptive discovery mechanisms (*beginning Phase II*)

- Complete scalable (up to 500 nodes) discrete-event simulation model of UPnP – based on source code from Intel’s Linux SDK for UPnP (*preparing our Phase II models for easy conversion to implementation during Phase III*)

- Extend our existing generic structural model of service discovery systems to cover behavior, message vocabulary, and consistency conditions (*we see this as a community service*)
Conclusions

- Emerging industry discovery protocols provide robustness properties through a division of responsibilities among: lower layer protocols, discovery protocols, and applications.

- Characterizing the behavior and robustness of commercial service discovery protocols is a necessary pre-condition to developing and evaluating adaptation mechanisms intended to improve the performance of such protocols.

- We described an approach to understand the behavior of service discovery protocols in the face of various faults: interface failure, message loss, and path and node failure. (To learn more about the approach, see: “Analyzing Properties and Behavior of Service Discovery Protocols Using an Architecture-based Approach”, C. Dabrowski and K. Mills, Proceedings of Working Conference on Complex and Dynamic Systems Architectures, sponsored by DARPA DASADA program)

- We applied the approach to characterize the performance of two different mechanisms (polling and eventing) for information propagation in various service discovery architectures (2-party and 3-party) and topologies (one and two SCMs) during interface failure.

- We are currently conducting characterizations of performance in the face of message loss and node failures.
Slides Containing Additional Details
Equating a Generic Structural Model of Service Discovery Architectures to Selected Commercial Discovery Systems

<table>
<thead>
<tr>
<th>Generic Model</th>
<th>Jini</th>
<th>UPnP</th>
<th>SLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service User</td>
<td>Client</td>
<td>Control Point</td>
<td>User Agent</td>
</tr>
<tr>
<td>Service Manager</td>
<td>Service or</td>
<td>Root Device</td>
<td>Service Agent</td>
</tr>
<tr>
<td></td>
<td>Device Proxy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Provider</td>
<td>Service</td>
<td>Device or Service</td>
<td>Service</td>
</tr>
<tr>
<td>Service Description</td>
<td>Service Item</td>
<td>Device/Service Description</td>
<td>Service Registration</td>
</tr>
<tr>
<td>Identity</td>
<td>Service ID</td>
<td>Universal Unique ID</td>
<td>Service URL</td>
</tr>
<tr>
<td>Type</td>
<td>Service Type</td>
<td>Device/Service Type</td>
<td>Service Type</td>
</tr>
<tr>
<td>Attributes</td>
<td>Attribute Set</td>
<td>Device/Service Schema</td>
<td>Service Attributes</td>
</tr>
<tr>
<td>User Interface</td>
<td>Service Applet</td>
<td>Presentation URL</td>
<td>Template URL</td>
</tr>
<tr>
<td>Program Interface</td>
<td>Service Proxy</td>
<td>Control/Event URL</td>
<td>Template URL</td>
</tr>
<tr>
<td>Service Cache Manger</td>
<td>Lookup Service</td>
<td>not applicable</td>
<td>Directory Service Agent (optional)</td>
</tr>
</tbody>
</table>
### The Six Combinations of Architecture, Topology, and Consistency-Maintenance Mechanism Used in Experiments

<table>
<thead>
<tr>
<th>Architectural Variant</th>
<th>Protocol Basis</th>
<th>Consistency-Maintenance Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Party</td>
<td>UPnP</td>
<td>Polling</td>
</tr>
<tr>
<td>Two-Party</td>
<td>UPnP</td>
<td>Notification (with notification registration on SCM)</td>
</tr>
<tr>
<td>Three-Party (Single SCM)</td>
<td>Jini</td>
<td>Polling (with service registration on SCM)</td>
</tr>
<tr>
<td>Three-Party (Single SCM)</td>
<td>Jini</td>
<td>Notification (with service registration and notification registration on SCM)</td>
</tr>
<tr>
<td>Three-Party (Dual SCM)</td>
<td>Jini</td>
<td>Polling (with service registration on SCM)</td>
</tr>
<tr>
<td>Three-Party (Dual SCM)</td>
<td>Jini</td>
<td>Notification (with service registration and notification registration on SCM)</td>
</tr>
</tbody>
</table>
### Specific Division of Failure-Recovery Responsibilities Used in Experiments

<table>
<thead>
<tr>
<th>Responsible Party</th>
<th>Recovery Mechanism</th>
<th>Two-Party Architecture (UPnP)</th>
<th>Three-Party Architecture (Jini)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lower-Layer Protocols</strong></td>
<td>UDP</td>
<td>No recovery</td>
<td>No recovery</td>
</tr>
<tr>
<td></td>
<td>TCP</td>
<td>Issue REX in 30-75 s</td>
<td>Issue REX in 30-75 s</td>
</tr>
<tr>
<td><strong>Discovery Protocols</strong></td>
<td>Lazy Discovery</td>
<td>SM: announces every 1800 s</td>
<td>SCM: announces every 120 s</td>
</tr>
<tr>
<td></td>
<td>Aggressive Discovery</td>
<td>SU: issues <code>Msearch</code> every 120 s (after purging SD)</td>
<td>SU and SM: issue seven probes (at 5 s intervals) only during startup</td>
</tr>
<tr>
<td><strong>Application Software</strong></td>
<td>Ignore REX</td>
<td>SU: <code>HTTP Get</code> <code>Poll</code> SM: Notification</td>
<td>SU: <code>FindService</code> <code>Poll</code> SCM: Notification</td>
</tr>
<tr>
<td></td>
<td>Retry in 120 s after REX</td>
<td>SU: <code>HTTP Get</code> after discovery (retry up to three times) <code>Subscribe</code> requests</td>
<td>SM: depositing or refreshing SD copy on SCM SU: registering and refreshing notification requests with SCM</td>
</tr>
<tr>
<td></td>
<td>Discard Knowledge</td>
<td>SU: purge SD after failure to receive SM announcement within 1800 s</td>
<td>SU and SM: purge SCM after 540 s of continuous REX</td>
</tr>
</tbody>
</table>
### Significant Parameters and Values Used in Experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polling interval</td>
<td>180 s</td>
</tr>
<tr>
<td>Registration TTL</td>
<td>1800 s</td>
</tr>
<tr>
<td>Time to retry after REX (if applicable)</td>
<td>120 s</td>
</tr>
<tr>
<td>Announce interval</td>
<td>1800 s</td>
</tr>
<tr>
<td>Msearch query interval</td>
<td>120 s</td>
</tr>
<tr>
<td>SU purges SD</td>
<td>At TTL expiration</td>
</tr>
<tr>
<td>Probe interval</td>
<td>5 s (7 times)</td>
</tr>
<tr>
<td>Announce interval</td>
<td>120 s</td>
</tr>
<tr>
<td>SM or SU purges SCM</td>
<td>After 540 s with only REX</td>
</tr>
<tr>
<td>Failure incidence</td>
<td>Once per run for each node</td>
</tr>
<tr>
<td>Failure scope</td>
<td>Transmitter, receiver, or both with equal likelihood</td>
</tr>
<tr>
<td>Failure duration</td>
<td>5% increments of 5400 s from 0 to 75%</td>
</tr>
<tr>
<td>UDP transmission delay</td>
<td>10 us constant</td>
</tr>
<tr>
<td>TCP transmission delay</td>
<td>10-100 us uniform</td>
</tr>
<tr>
<td>Per-item processing delay</td>
<td>100 us for cache items, 10 us for other items</td>
</tr>
</tbody>
</table>
Console Output from a Sample Experiment Run

Rate - 5
Run number - 21

SM 1 OUT Interface       down 365, up 635
SCM 1 OUT Interface       down 2417, up 2687
SCM 2 IN & OUT Interface  down 519, up 789
SU 1  IN Interface     down 2238, up 2508
SU 2  IN Interface    down 3256, up 3526
SU 3  IN Interface   down 207,  up 477
SU 4  OUT Interface down 2876, up 3146
SU 5  IN Interface down 4478, up 4748

Performance:

SM  1 346.00000 346.00000 6 17
SCM 1 346.00000 346.00016 61 102
SCM 2 346.00000 346.00015 61 105
SU  1 346.00000 346.00109 0 11
SU  2 346.00000 346.00109 0 11
SU  3 346.00000 5400.00000 4 11
SU  4 346.00000 346.00109 0 11
SU  5 346.00000 346.00114 0 11
Update Responsiveness ($R$)

Let $D$ be a deadline by which we wish to propagate information to each service user (SU) node ($n$) in a service discovery topology.

Let $t_C$ be the creation time of the information that we wish to propagate, where $t_C < D$.

Let $t_{U(n)}$ be the time that the information is propagated to SU $n$, where $n = 1$ to $N$, and $N$ is the total number of SUs in a service discovery topology.

Define information-propagation latency ($L$) for an SU $n$ as:

$$L_n = \frac{(t_{U(n)} - t_C)}{(\max(D, t_{U(n)}) - t_C)}.$$

Define update responsiveness ($R$) for an SU $n$ as:

$$R_n = 1 - L_n.$$
Let the definitions related to Update Responsiveness, $R$, hold.

Let $X$ represent the number of runs during which a particular service discovery topology is observed under identical conditions.

Recalling that $N$ is the total number of SUs in a service discovery topology, define the number of SUs observed under identical conditions as:

$$O = X \cdot N.$$  

Define the probability of failure to propagate information to an SU as:

$$P(F) = \frac{\text{count}(R_{i,j} == 0)}{O}, \text{ where } i = 1..N \text{ and } j = 1..X.$$  

Define the Update Effectiveness for a given set of conditions as:

$$U = 1 - P(F).$$
Update Efficiency ($E$)

Let the preceding definitions associated with Update Responsiveness and Update Effectiveness hold.

Let $M$ be the minimum number of messages needed to propagate information from all SMs to all SUs.

Let $S$ be the observed number of messages sent while attempting (failures may occur) to propagate information from all SMs to all SUs in a given run of the topology.

Define average Update Efficiency as:

$$E_{avg} = \frac{\text{sum}(M/S_k)}{X}, \text{ where } k = 1..X.$$
## Summary Statistics for Performance of Each Combination on Each Metric

<table>
<thead>
<tr>
<th>Combination</th>
<th>Median Responsiveness</th>
<th>Effectiveness</th>
<th>Average Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Party Notification</td>
<td>0.663</td>
<td>0.921</td>
<td>0.212</td>
</tr>
<tr>
<td>Two-Party Polling</td>
<td>0.615</td>
<td>0.973</td>
<td>0.251</td>
</tr>
<tr>
<td>Three-Party Notification (Single SCM)</td>
<td>0.601</td>
<td>0.894</td>
<td>0.389</td>
</tr>
<tr>
<td>Three-Party Polling (Single SCM)</td>
<td>0.530</td>
<td>0.911</td>
<td>0.201</td>
</tr>
<tr>
<td>Three-Party Notification (Dual SCM)</td>
<td>0.655</td>
<td>0.942</td>
<td>0.221</td>
</tr>
<tr>
<td>Three-Party Polling (Dual SCM)</td>
<td>0.587</td>
<td>0.927</td>
<td>0.110</td>
</tr>
</tbody>
</table>
## 95% C.I. for Each Metric-Combination at Selected Failure Rates

<table>
<thead>
<tr>
<th></th>
<th>Responsiveness</th>
<th></th>
<th></th>
<th>Effectiveness</th>
<th></th>
<th></th>
<th>Efficiency</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5%</td>
<td>40%</td>
<td>75%</td>
<td>5%</td>
<td>40%</td>
<td>75%</td>
<td>5%</td>
<td>40%</td>
<td>75%</td>
<td></td>
</tr>
<tr>
<td>Two-Party Notification</td>
<td>1.000</td>
<td>0.561</td>
<td>0.111</td>
<td>0.970</td>
<td>0.954</td>
<td>0.709</td>
<td>0.354</td>
<td>0.065</td>
<td>0.031</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>0.783</td>
<td>0.162</td>
<td>0.977</td>
<td>0.966</td>
<td>0.787</td>
<td>0.467</td>
<td>0.220</td>
<td>0.354</td>
<td></td>
</tr>
<tr>
<td>Two-Party Polling</td>
<td>0.975</td>
<td>0.501</td>
<td>0.076</td>
<td>1.000</td>
<td>0.993</td>
<td>0.760</td>
<td>0.501</td>
<td>0.031</td>
<td>0.042</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.980</td>
<td>0.849</td>
<td>0.138</td>
<td>1.000</td>
<td>0.993</td>
<td>0.826</td>
<td>0.666</td>
<td>0.230</td>
<td>0.059</td>
<td></td>
</tr>
<tr>
<td>Three-Party Notification (Single SCM)</td>
<td>1.000</td>
<td>0.605</td>
<td>0.042</td>
<td>0.993</td>
<td>0.939</td>
<td>0.521</td>
<td>0.827</td>
<td>0.099</td>
<td>0.033</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td>0.095</td>
<td>0.993</td>
<td>0.955</td>
<td>0.652</td>
<td>1.000</td>
<td>0.504</td>
<td>0.320</td>
<td></td>
</tr>
<tr>
<td>Three-Party Polling (Single SCM)</td>
<td>0.974</td>
<td>0.244</td>
<td>0.043</td>
<td>1.000</td>
<td>0.946</td>
<td>0.660</td>
<td>0.387</td>
<td>0.043</td>
<td>0.040</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.980</td>
<td>0.412</td>
<td>0.083</td>
<td>1.000</td>
<td>0.960</td>
<td>0.753</td>
<td>0.512</td>
<td>0.173</td>
<td>0.164</td>
<td></td>
</tr>
<tr>
<td>Three-Party Notification (Dual SCM)</td>
<td>1.000</td>
<td>0.562</td>
<td>0.099</td>
<td>0.970</td>
<td>0.977</td>
<td>0.730</td>
<td>0.335</td>
<td>0.035</td>
<td>0.009</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.000</td>
<td>1.000</td>
<td>0.143</td>
<td>0.977</td>
<td>0.983</td>
<td>0.803</td>
<td>0.599</td>
<td>0.290</td>
<td>0.096</td>
<td></td>
</tr>
<tr>
<td>Three-Party Polling (Dual SCM)</td>
<td>0.974</td>
<td>0.391</td>
<td>0.056</td>
<td>1.000</td>
<td>0.939</td>
<td>0.660</td>
<td>0.218</td>
<td>0.033</td>
<td>0.019</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.986</td>
<td>0.543</td>
<td>0.096</td>
<td>1.000</td>
<td>0.955</td>
<td>0.753</td>
<td>0.273</td>
<td>0.103</td>
<td>0.059</td>
<td></td>
</tr>
</tbody>
</table>