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

Kevin Mills, Doug Montgomery, Scott Rose,
Stephen Quirolgico, and Chris Dabrowski
NIST

Mackenzie Britton (SMU) and Kevin Bowers (RPI & CMU)
NSF Summer University Research Fellows

DARPA FTN PI Meeting
July 22, 2003

Self-Organizing Systems for Hostile & Volatile Environments

Presentation Outline

- One-Page Review of Project Objective and Plan
- One-Page Refresher on Service-Discovery Protocols
- Summary of Accomplishments on the Project (to date)
- An Autonomic Failure-Detection Algorithm for Distributed Systems
  - Applied to Service Registration in the Service Location Protocol (SLP)
  - Applied to Jini Leasing **(SEE DEMO on WEDNESDAY EVENING)**
- Performance of Service-Discovery Systems under Node Failure
- Plan for Final Six Months
- Conclusions

Project Objective

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

Project Plan – Three Tasks

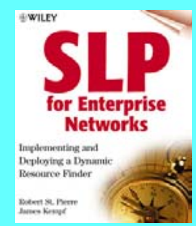
- TASK I – characterize the fault-tolerance and performance of selected service discovery protocols [Universal Plug-and-Play (UPnP), Service Location Protocol (SLP) and Jini] as specified and implemented
  - develop simulation models for each protocol
  - establish performance benchmarks based on default or recommended parameter values and on required or most likely implementation of behaviors
- TASK 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
- TASK III – implement and validate the most promising algorithms in publicly available reference software

Service-Discovery Protocols in Essence

Dynamic multi-party protocols that enable *distributed services*:

(1) to *discover* each other without prior arrangement,
(2) to *describe* 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 topology.

Selected First-Generation Service-Discovery Protocols

JINI — 3-Party Design	UPnP FORUM — 2-Party Design	 Adaptive 2/3-Party Design
The Salutation Consortium — Vertically Integrated 3-Party Design	HAVi — Network-Dependent 3-Party Design	Bluetooth™ — Network-Dependent 2-Party Design

Summary of Accomplishments on the Project (as of July 2003)

TASK I

- Developed and publicly released simulation models for Jini, UPnP, and SLP
  - SLX™ discrete-event simulations for Jini, UPnP, and SLP
  - Rapide simulations for Jini and UPnP

- Characterized response of Jini and UPnP to various types of failure
  - Node Failure: "Performance of Service-Discovery Architectures in Response to Node Failures", *Proceedings of SERP'03*, June 2003, CSREA, pp. 95-101.
  - Communications Failure: "Understanding Consistency Maintenance in Service Discovery Architectures during Communication Failure", *Proceedings of 3rd International Workshop on Software Performance*, ACM, July 2002, pp. 168-178.
  - Message Loss: "Understanding Consistency Maintenance in Service Discovery Architectures in Response to Message Loss", *Proceedings of the 4th International Workshop on Active Middleware Services*, IEEE Computer Society, July 2002, pp. 51-60.
  - Power Failure Restart: "Analyzing Properties and Behavior of Service Discovery Protocols using an Architecture-based Approach", *Proceedings of DARPA Working Conference on Complex and Dynamic Systems Architecture*, December 2001.
  - Self-Healing: "Understanding Self-healing in Service Discovery Systems", *Proceedings of ACM SigSoft Workshop on Self-healing Systems*, November 2002, pp. 15-20.

- Characterized failure response in SLP for communications failure, message loss, and power-failure restart (*currently working on node-failure case*)

Summary of Accomplishments on the Project (as of July 2003)

TASK II

- Designed algorithms to automatically self-regulate performance of various service-discovery functions
  - Adaptive Jitter-Control Algorithm for Multicast Search in UPnP
  - Autonomic Failure-Detection Algorithm (including analysis) – MORE ON THIS AHEAD
  - Self-adaptive Inverted Leasing Algorithm for Jini (including analysis)

- Developed and publicly released SLX™ discrete-event simulation models
  - Adaptive Jitter Control Algorithm for UPnP M-Search
  - Autonomic Leasing Algorithm for Jini – MORE ON THIS AHEAD
  - Inverted Leasing Algorithm for Jini
  - Autonomic Service Registration and Refresh Algorithm for SLP – MORE ON THIS AHEAD

- Published algorithms and performance characterizations
  - UPnP M-Search: "Adaptive Jitter Control for UPnP M-Search", *Proceedings of IEEE ICC 2003*, May 2003.
  - Self-Adaptive Leasing for Jini: "Self-adaptive Leasing for Jini", *Proceedings of IEEE PerCom 2003*, March 2003.
  - Improving Failure Responsiveness: "Improving Failure Responsiveness in Jini Leasing", *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition* (DISCEX-III 2003), IEEE Computer Society, April 2003, Vol. II, pp. 103-105.
  - Self-Management: "Self-Managed Leasing for Distributed Systems", Poster Paper in the *Proceedings of the 1st Workshop on Algorithms and Architectures for Self-Managing Systems*, co-sponsored by ACM SIGMETRICS, June 2003.

Summary of Accomplishments on the Project (as of July 2003)

TASK III

- Implemented Autonomic Lease-Granting Algorithm for Jini
  - Modified Jini Lookup Service code (publicly released by Sun Microsystems)
  - Modified Jini Lookup-Service Administrative Interface to input policy parameters
  - Implemented test system software and infrastructure to generate, control, and monitor thousands of Jini services

- Demonstrated Autonomic Leasing for Jini
  - DISCEX III in April 2003 (Washington, D.C.)
  - Self-Managing Systems Workshop in June 2003 (San Diego, CA)
  - **FTN PI Meeting in July 2003 (Honolulu, HI) – PLEASE STOP BY WEDNESDAY EVENING**

- Validated Autonomic Leasing Algorithm for Jini – MORE ON THIS AHEAD
  - Deployed modified Jini Lookup Service and test system and conducted controlled experiments, collecting data for analysis
  - Implemented an analytical model of the autonomic leasing algorithm and evaluated it with the same parameters used in the live experiments
  - Iterated the Jini autonomic leasing simulation model with the same parameters used in the live experiments
  - Compared results – correspondence quite good among the measured, simulated, and analytical results

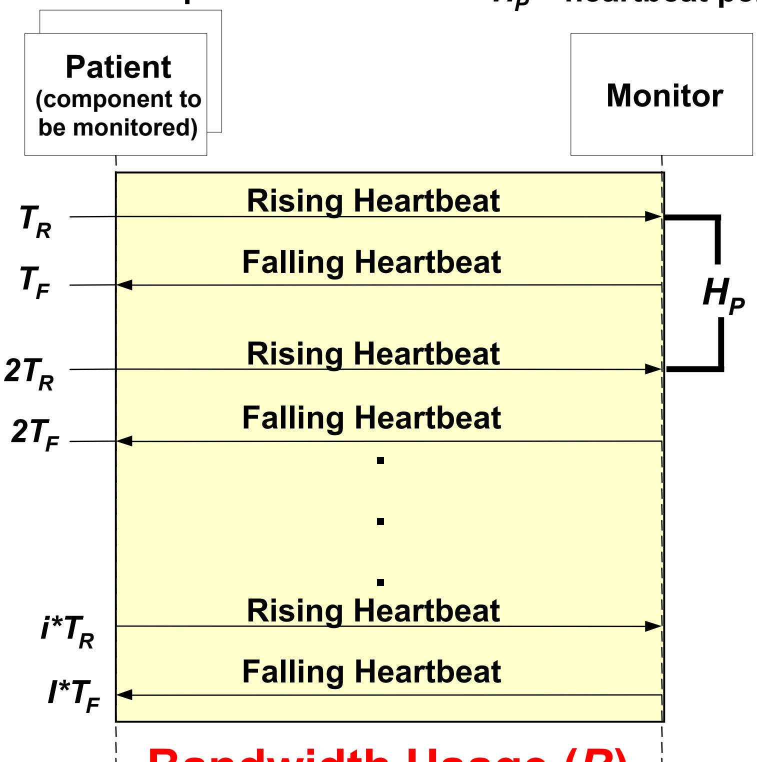
Presentation Outline

- One-Page Review of Project Objective and Plan
- One-Page Refresher on Service-Discovery Protocols
- Summary of Accomplishments on the Project (to date)
- An Autonomic Failure-Detection Algorithm for Distributed Systems
  - Applied to Service Registration in the Service Location Protocol (SLP)
  - Applied to Jini Leasing **(SEE DEMO on WEDNESDAY EVENING)**
- Performance of Service-Discovery Systems under Node Failure
- Plan for Final Six Months
- Conclusions

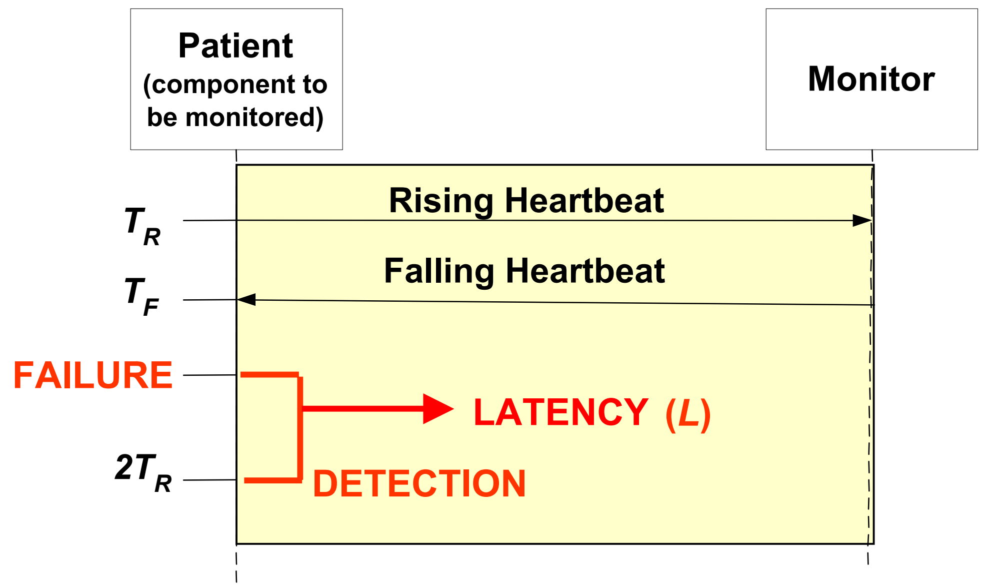
Heartbeat-Based Failure-Detection in Distributed Systems

N = number of patients

H_P = heartbeat period



Bandwidth Usage (B)



The faster the heartbeat the smaller the failure-detection latency, but the larger the bandwidth consumption.

For a given heartbeat rate and message size, the larger the patient population, the greater the bandwidth consumption.

S_R = size of Rising Heartbeat Message
S_F = size of Falling Heartbeat Message

An Autonomic Failure-Detection Algorithm for Distributed Systems

Goal: limit bandwidth usage to B_A and assure avg. worst-case failure-detection latency (L_{WORST}), while achieving better avg. failure-detection latency $L < L_{WORST}$ when $N < N_{MAX}$

Analysis

$H_{MAX} = 2L_{WORST}$ Avg. worst-case failure-detection latency determines maximum heartbeat period

$C = B_A / (S_R + S_F)$ Allocated bandwidth B_A and size of rising S_R and falling S_F heartbeat messages determine system capacity in heartbeats per second

$H_{MIN} = 1/C$ Assuming minimum system size of 1, C determines minimum heartbeat period

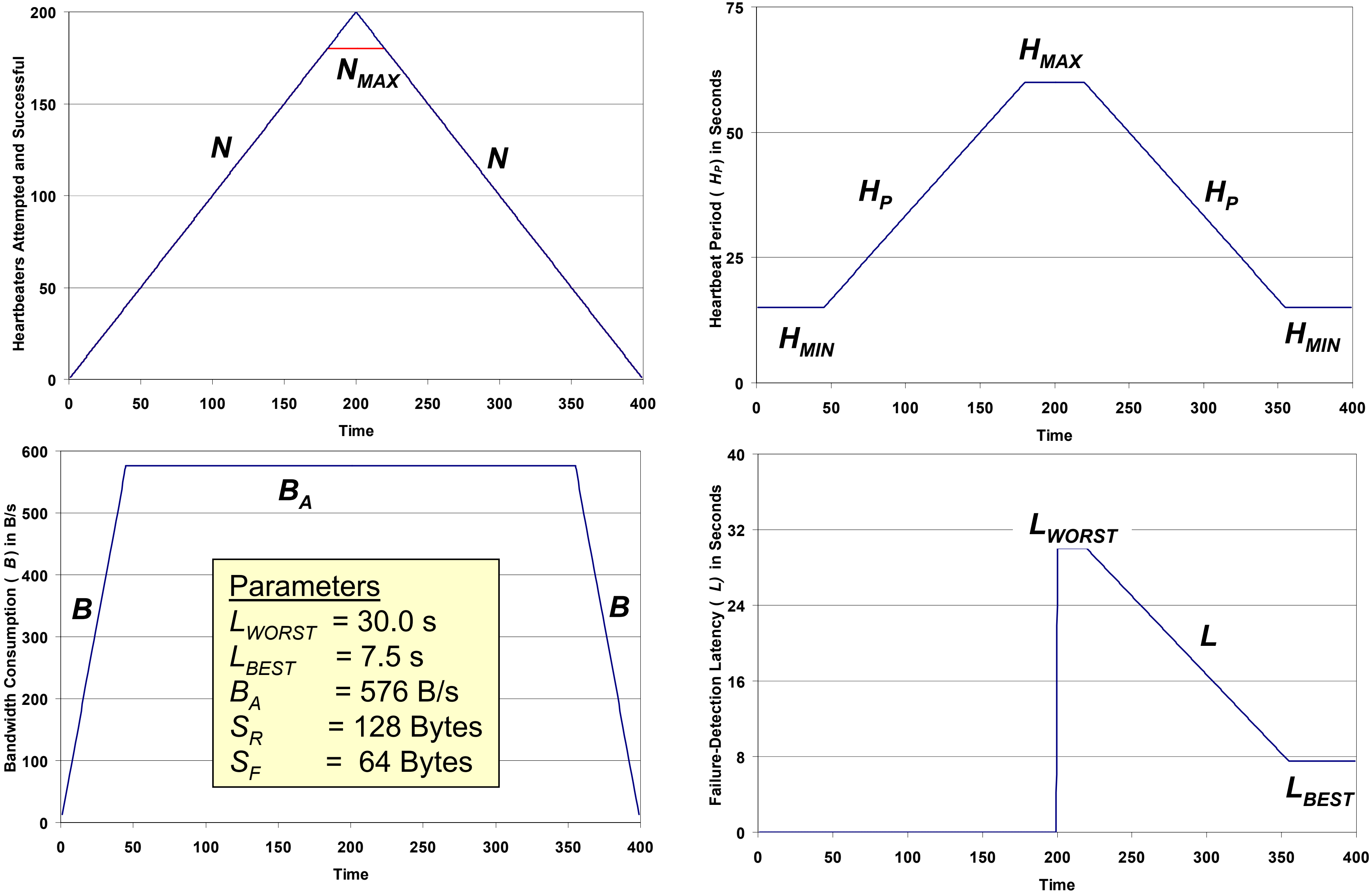
$H_{MIN} = 2L_{BEST}$ However, $1/C$ might place too great a load on an individual heart, so instead choose a avg. best-case failure-detection latency to determine minimum heartbeat period

$H_{MIN} \leq H_P \leq H_{MAX}$ Vary the heartbeat period within this range, using the following algorithm

Autonomic Algorithm for Varying H_P

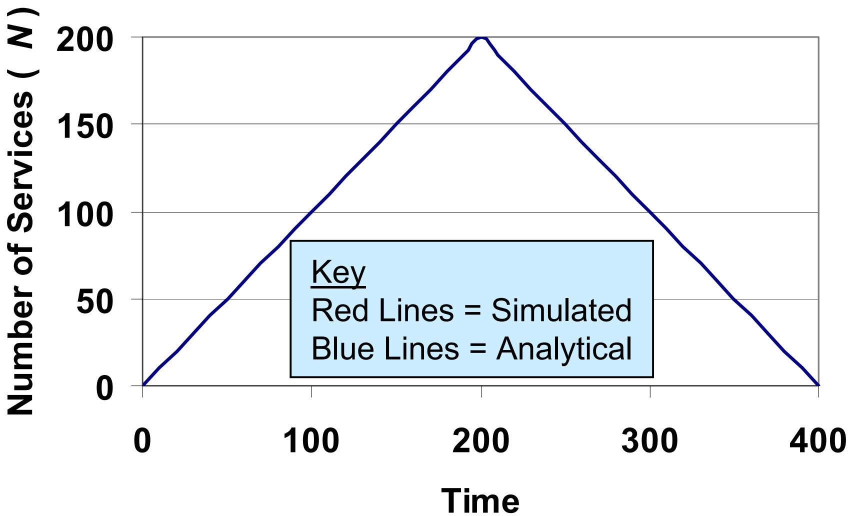
set $H_P = N / C$;
if $H_P > H_{MAX}$
  then refuse to monitor the heartbeat;
  elseif $H_P < H_{MIN}$
    then set $H_P = H_{MIN}$;
  endif
endif

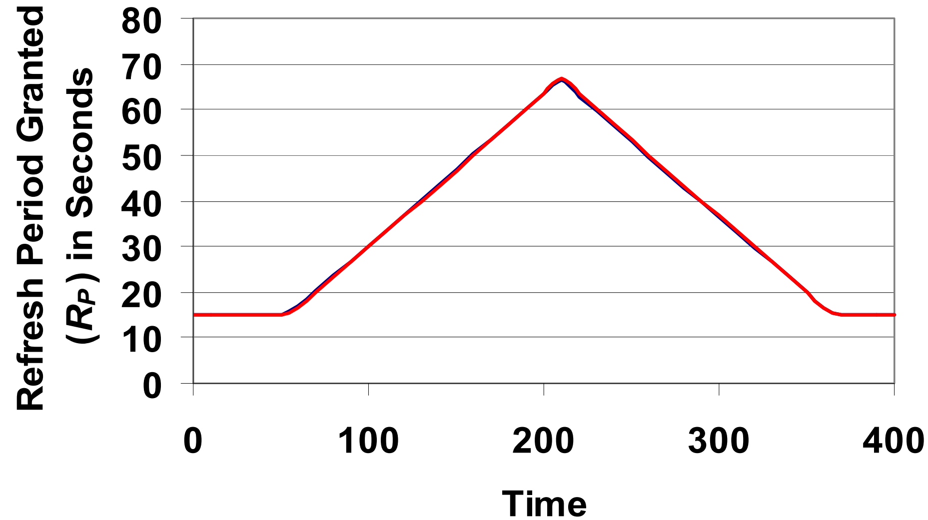
Analysis of Autonomic Heartbeat Algorithm in Operation

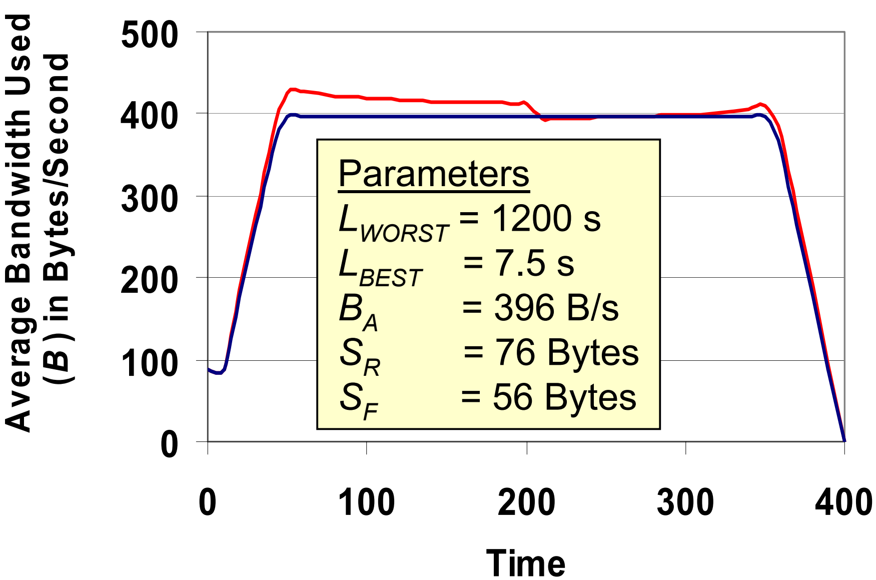


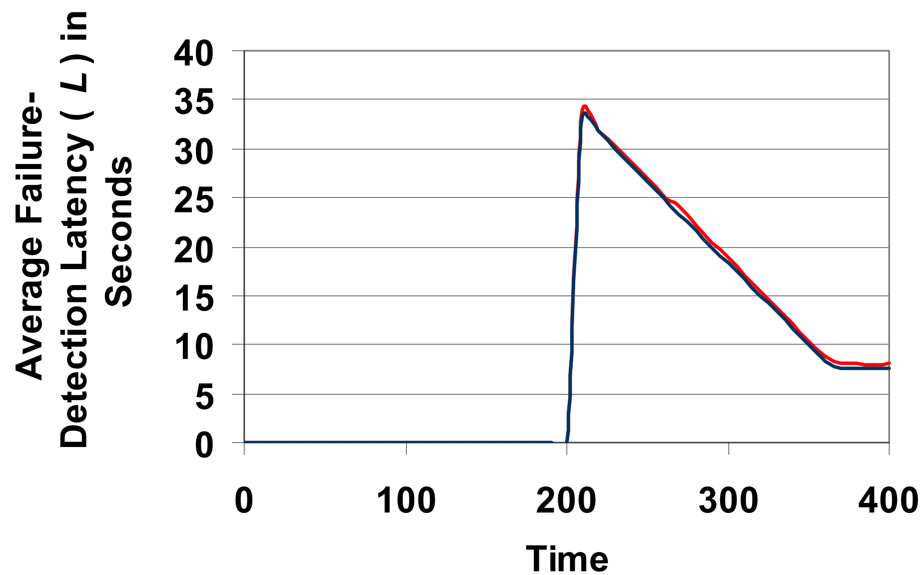
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Autonomic Heartbeat Algorithm Applied to SLP Service Registration & Refresh



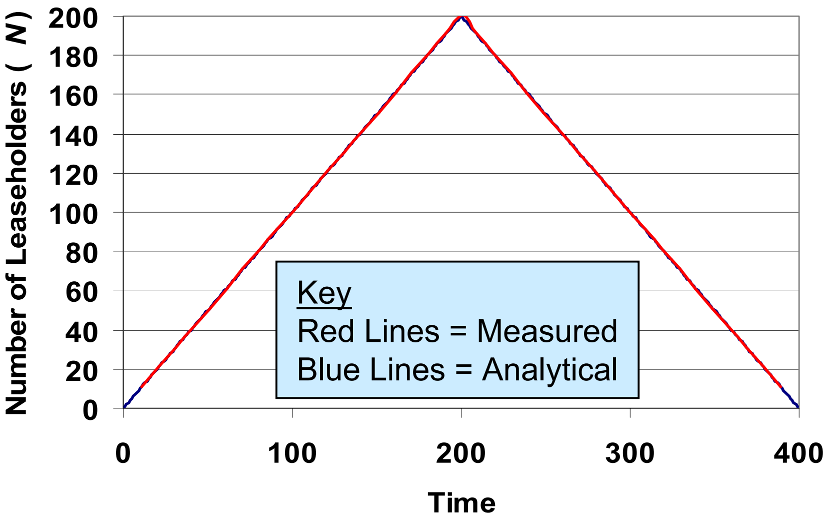


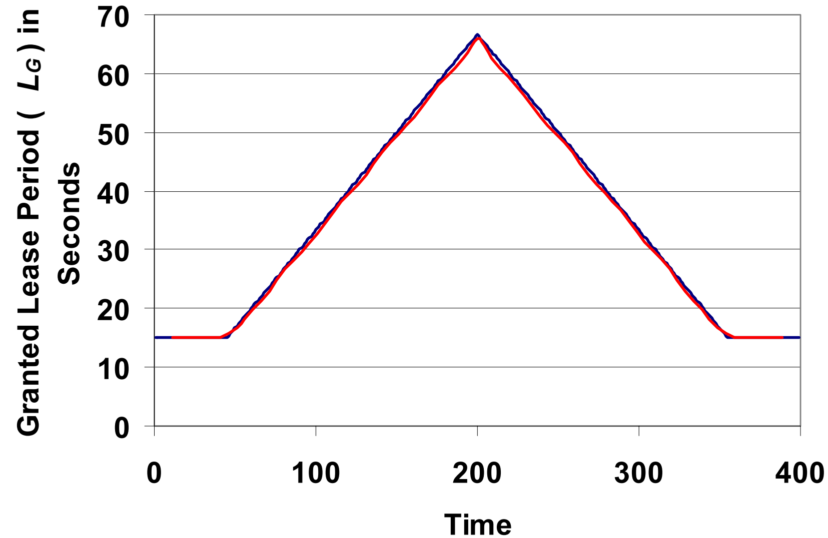


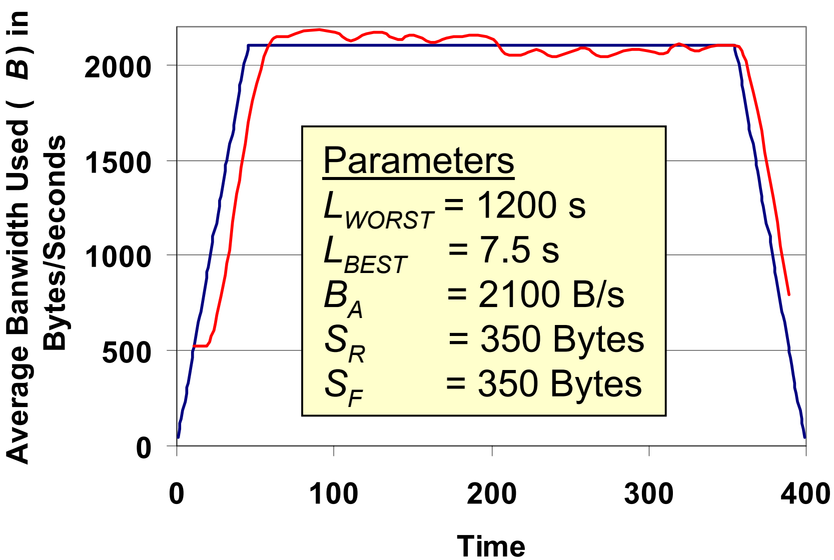


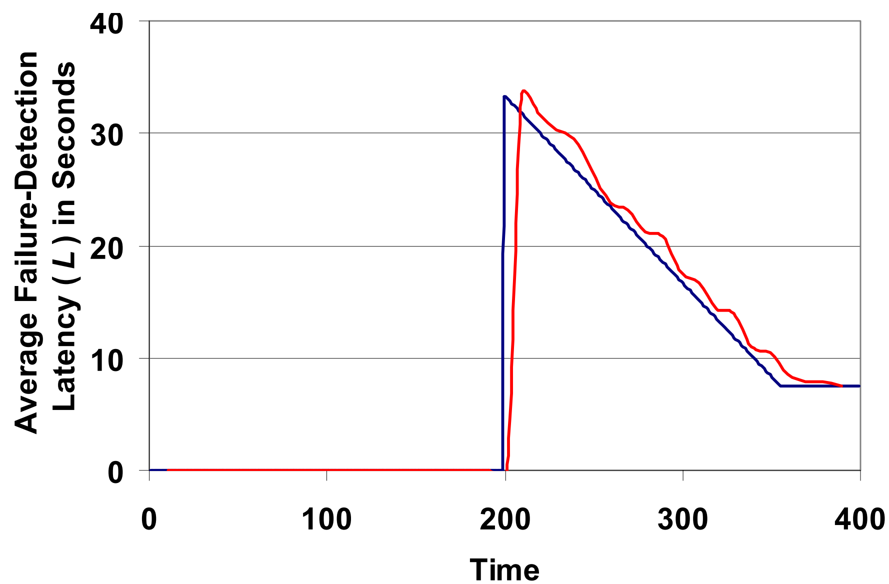
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Autonomic Heartbeat Algorithm Applied to Jini Leasing (SEE WEDS. DEMO)





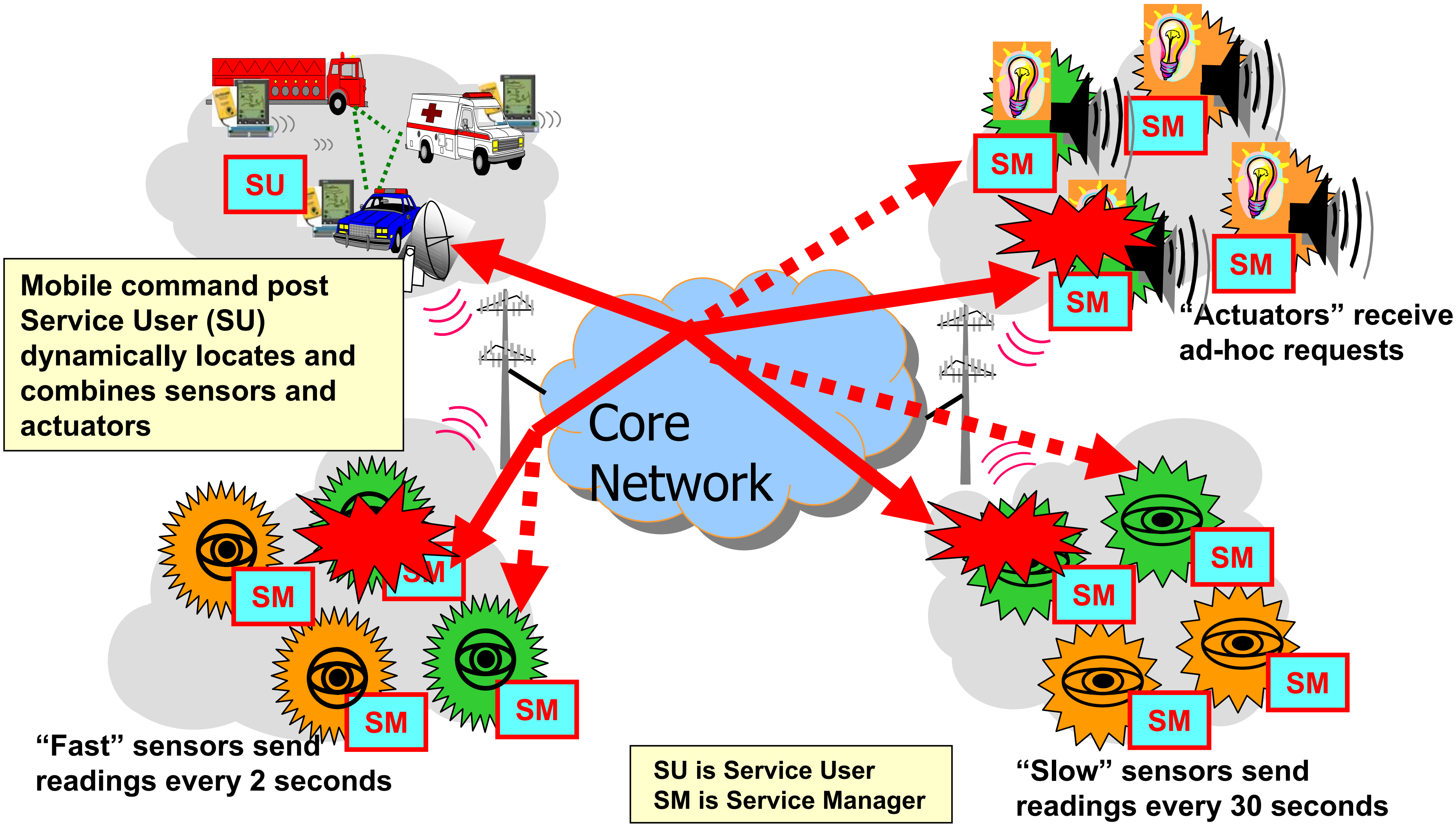




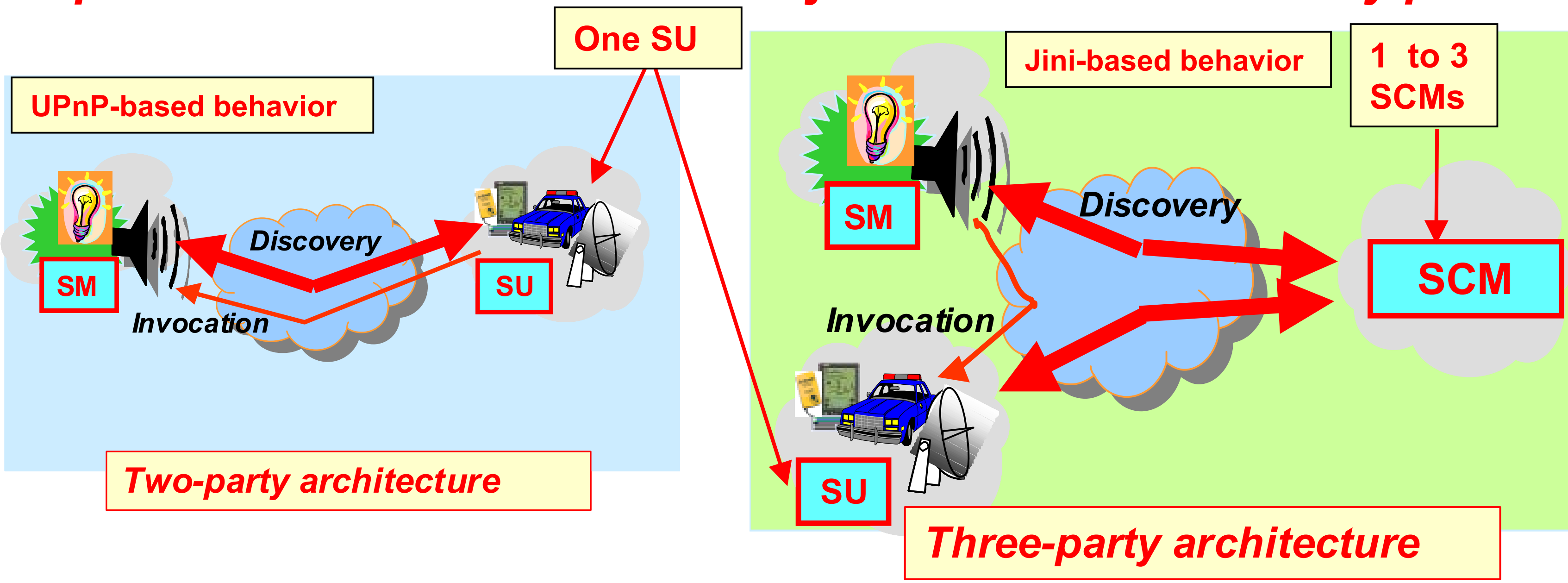
Presentation Outline

- One-Page Review of Project Objective and Plan
- One-Page Refresher on Service-Discovery Protocols
- Summary of Accomplishments on the Project (to date)
- An Autonomic Failure-Detection Algorithm for Distributed Systems
  - Applied to Service Registration in the Service Location Protocol (SLP)
  - Applied to Jini Leasing **(SEE DEMO on WEDNESDAY EVENING)**
- Performance of Service-Discovery Systems under Node Failure
- Plan for Final Six Months
- Conclusions

How Well Do Service-Discovery Protocols Replace Services Lost to Node Failure?



Compared two architectures used by most service discovery protocols



- In 2-party architecture, SU discovers SMs through multicast search strategy
  - SU registers for notification of change in status of service (renewed every 300s)
- In 3-party-party architecture, both SMs and SUs discover SCMs; SU obtains services through SCM intermediary
  - SMs register services (renewed every 300s for fast sensors and every 60s for slow sensors and actuators); SU registers notification requests (renewed every 300s)
- SU detects failure of services through (1) non-response or (2) notification of registration renewal failure (heartbeat mechanism). Upon loss of service……
  - 2-party SU multicasts queries to SMs every 120s
  - 3-party SU queries SCMs for service; If SCMs lost, SU (and SMs) listen for SCM announcements (every 120s)

Experiment Design

Goal of SU is to be functional; i.e, to continually possess one instance of each type of service (“fast” sensor, “slow” sensor, & actuator).

- **When >= 1 type of sensor is missing, SU is non-functional**
- **To focus on alternative architectures & associated processes, mechanisms such as service caching factored out**

Q = end of quiescent period (60 s)
D = end of experiment (1800 s)
R = failure rate (variable from 0% - 80% in 10% increments)

0 **Q** **D**

One or more failures for each node via a Mean-Time-to-Failure

Initial Discovery occurs

SMs and SCMs fail randomly using a “stepped” normal distribution (3 steps with *P*=.15, .7, and .15) around *MTF* = (1-*R*)D*. Three failure durations: short (*P*=.1), medium (*P*=.7), long (*P*=.2). SU doesn’t fail.**

TIME

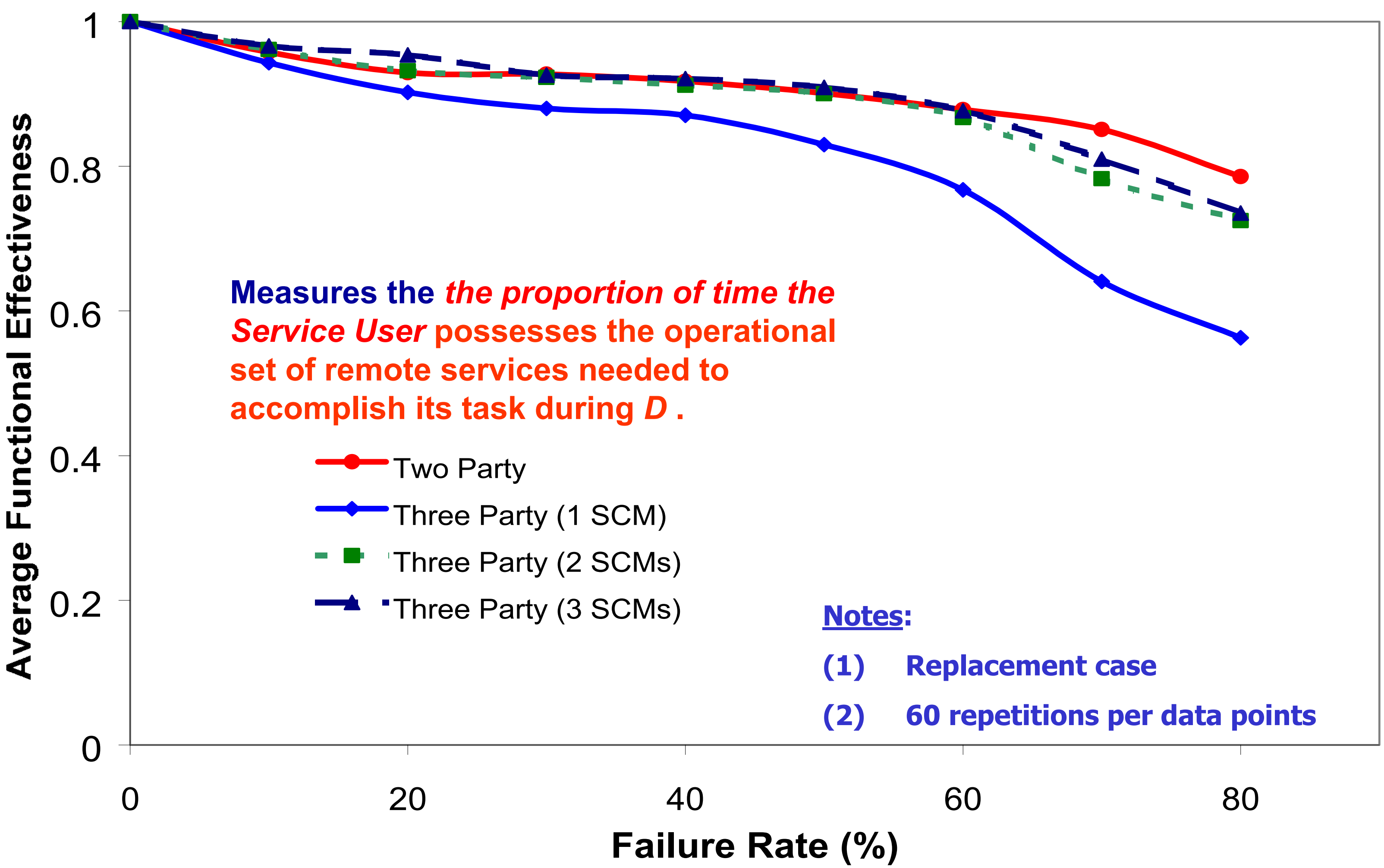
Variations Considered (but not all results reported here):

(1) After a failure period, a failed node could either be Replaced (i.e, cached data does not persist) or Restored (i.e., cached data may persist)

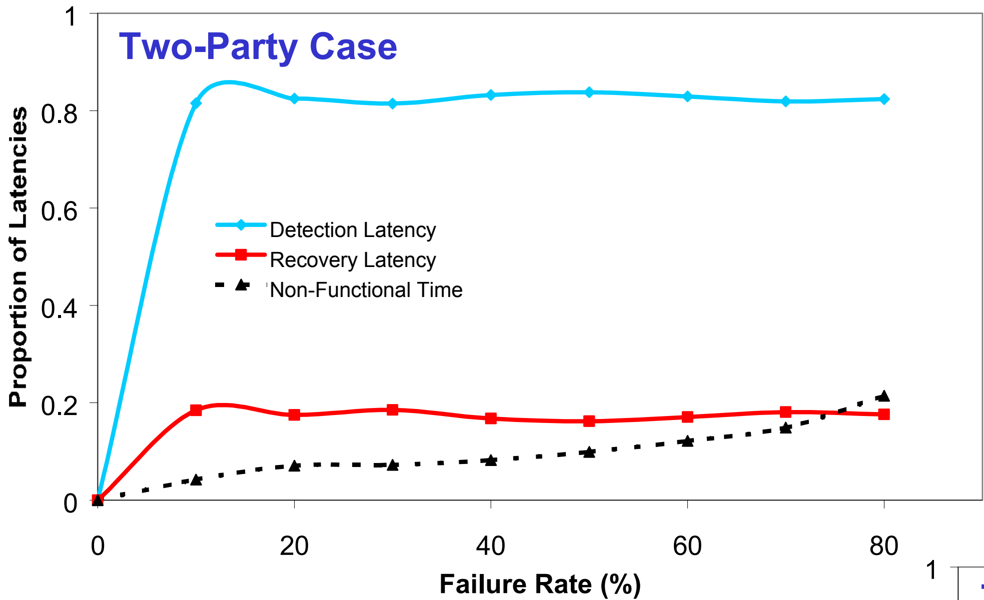
(2) At least one SM of each type could be required to remain operational or all SMs could be allowed to fail

(3) All SCMs are always allowed to fail

Functional Effectiveness of Two-Party vs. Three-Party When One SM of Each Type is Always Available

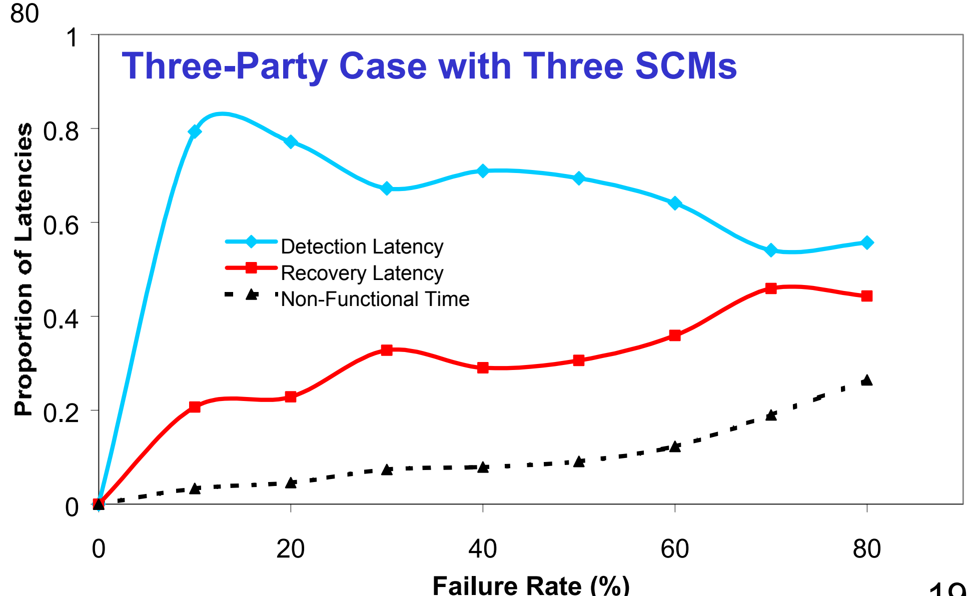


Non-Functional Time Decomposed Proportionately into Detection Latency and Recovery Latency

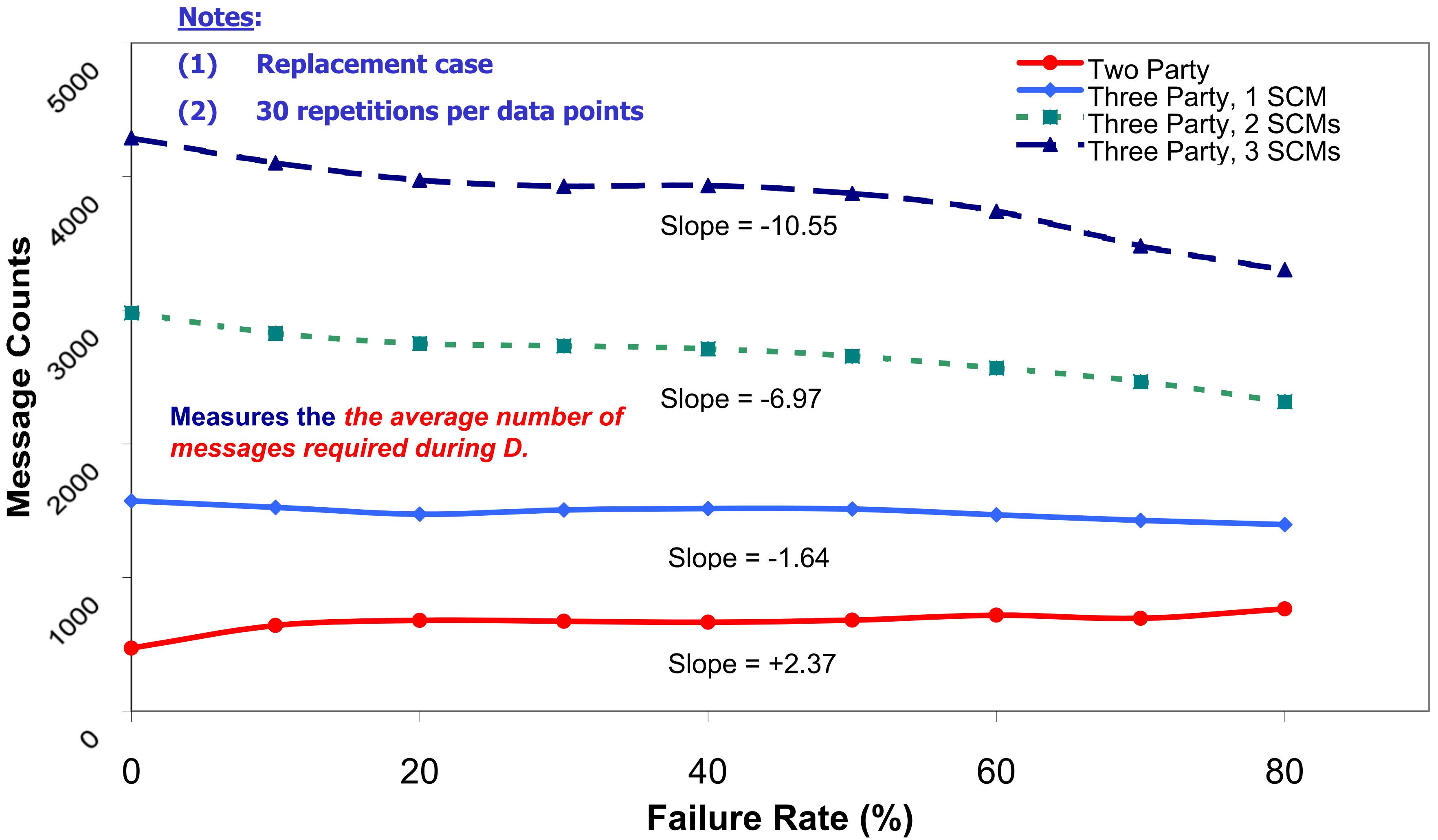


<u>Two-Party Architecture</u>:
Failure-Detection Latency dominates at all failure rates.

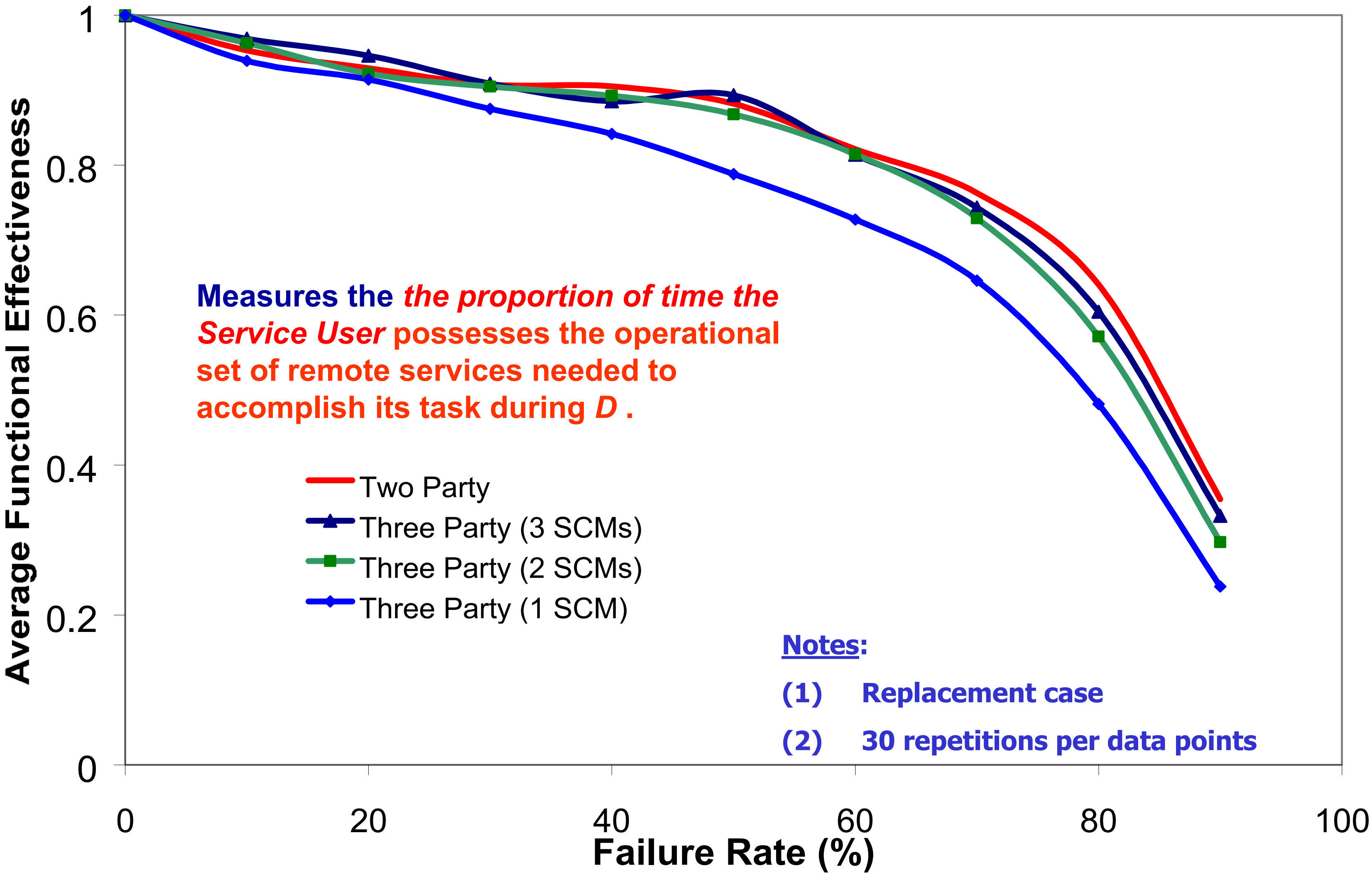
<u>Three-Party Architecture (3 SCMs)</u>:
Failure-Detection Latency and Recovery Latency come closer as Failure Rate increases because SCMs are required for rendezvous, but can all be failed.



Efficiency of Two-Party vs. Three-Party When One SM of Each Type is Always Available



Functional Effectiveness of Two-Party vs. Three-Party When All SMs of Each Type Can Fail



Presentation Outline

- One-Page Review of Project Objective and Plan
- One-Page Refresher on Service-Discovery Protocols
- Summary of Accomplishments on the Project (to date)
- An Autonomic Failure-Detection Algorithm for Distributed Systems
  - Applied to Service Registration in the Service Location Protocol (SLP)
  - Applied to Jini Leasing **(SEE DEMO on WEDNESDAY EVENING)**
- Performance of Service-Discovery Systems under Node Failure
- Plan for Final Six Months
- Conclusions

Plan for the Final Six Months

ONE IMPLEMENTATION, THREE CONFERENCE PAPERS, AND TWO JOURNAL PAPERS

- Implement our autonomic failure-detection algorithm for registration refresh and for directory polling in the *meshSLP* implementation available from Columbia University
- Write and submit a paper to WOSP 2004 on "An Autonomic Failure-Detection Algorithm for Distributed Systems"
- Write and submit two conference papers on failure-response in SLP (one covering message loss and communication failure and one covering node failure and power failure restart)
- Formalize a generic model of service-discovery architectures, including structure, behavior, and properties – write and submit a journal paper
- Write and submit a journal paper that characterizes the failure response of three service discovery protocols: Jini, UPnP, and SLP under hostile and volatile conditions

Conclusions

BY PROJECT'S END:

- We will have **characterized performance and failure response for the three most widely** accepted first-generation service **discovery protocols** (UPnP, SLP, and Jini), published our findings, and released the simulation models we used and data we collected

- We will have **devised and investigated three self-adaptive algorithms**: autonomic failure detection (applied to various aspects of service-discovery protocols), inverted leasing, and adaptive jitter control for multicast search, published our findings, and released the simulation models we used and data we collected

- We will have **implemented, demonstrated, and validated our autonomic failure detection algorithm**, as applied to Jini leasing, to SLP service registration and refresh, and to polling in Jini and SLP, published our findings, and released the implementations we used and data we collected

- We will have **constructed, implemented, tested, and verified a formal** generic **model** of service **discovery protocols** that:
  -- encompasses all the functions and features of Jini, UPnP, and SLP
  -- defines consistency conditions that such protocols should satisfy
  -- identifies missing functions and other weaknesses in existing service-discovery systems and proposes improvements
  -- incorporates the self-adaptive algorithms we developed