Cost-Effective Certification of High-Assurance Cyber Physical Systems

Kurt Rohloff
krohloff@bbn.com
BBN Technologies

Most Important Challenges and Needs

- Need dynamic behavior in high-confidence systems, especially with dynamic resource management.
  - Distributed System Interactions:
    - Multi-level Quality of Service (QoS).
    - Peer-to-Peer (P2P) interactions.
    - Interleaved Reconfiguration.
  - System timing issues:
    - Multi-time scale behavior with time-critical operations.
    - Mixed synchronous and asynchronous behaviors.
  - Lifecycle issues.
    - How can systems be quickly recertified?
    - Don't want to restart full process.

- Scalable techniques needed to certify composed network centric systems.
  - Conflicts in shared resource usage cause loss of composed certifiability.
- Elements of architecture, design, algorithms, analysis, simulation, testing and instrumentation/logging will all play a role.
  - Intelligently link augmentations of these elements together.

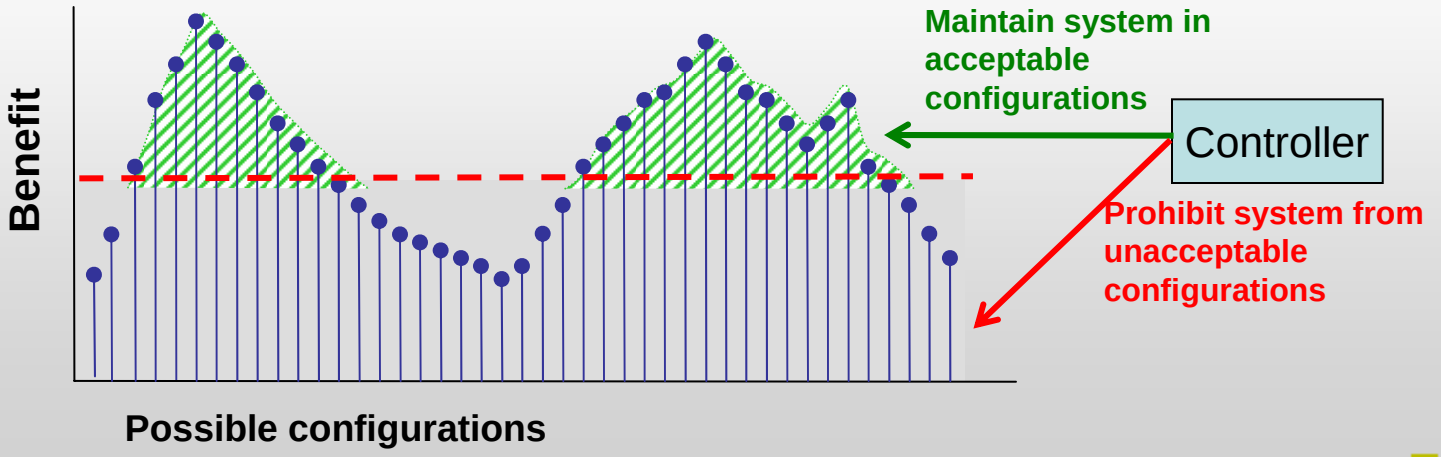
BBN TECHNOLOGIES

Deficiencies and Motivation

- Exhaustive testing, documentation, code review, formal methods for high-confidence software.
  - No longer economically feasible for highly complex, dynamic, distributed systems.
  - Inherent problems due to state explosion.
- A particular stumbling block is absence of methodology for dealing with dynamic resource management.
  - Current methods assume static allocations.

Restrict Operation to Certifiable Configurations

- Through the use of common middleware infrastructure and utility metric, we want to permit "certifiable" behavior to occur and prevent the system from entering into an "unacceptable" configurations.
- Important considerations:
  - Difficult to predict the effects of control operations in real-time.
  - May need to maintain a list of "fail-safe" default configurations.



Talk Overview

- A Generalizable Cyber-Physical Case-Study:

  The *ARMS* Program

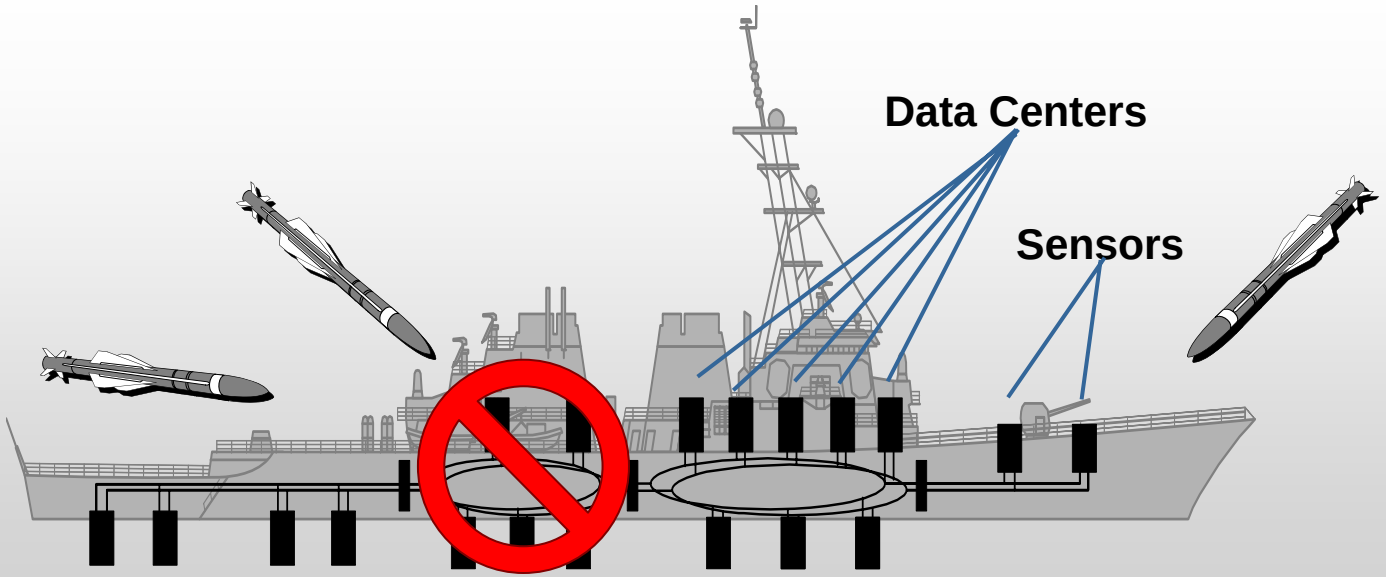
  - Distributed resource management for a Distributed Real-time Embedded (DRE) system.
  - Issues in *certification* of such a system.
- We use a *utility function* as a quality measure.
  - Utility function used as a *feedback control* signal.
- Want to use utility measurements as artifact for evidence-based certification.
  - Measures properties relevant to certification.

BBN TECHNOLOGIES

Context

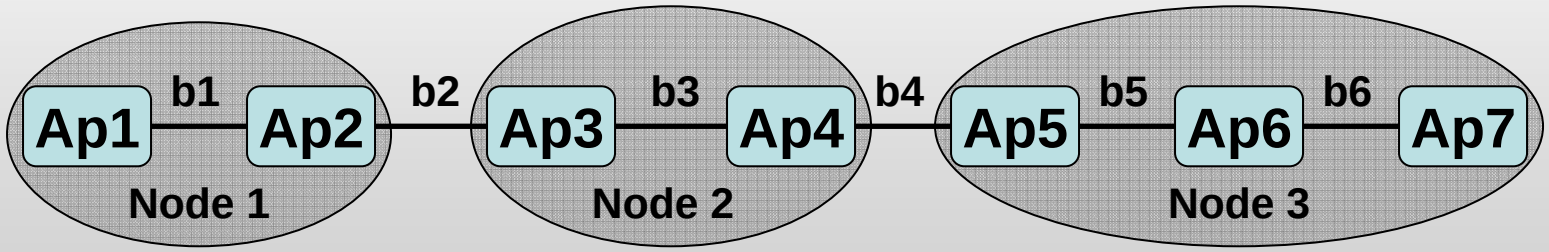
ARMS (Adaptive and Reflective Middleware System) program.

- Focus on developing a distributed computing environment that can rapidly respond to changing operating conditions.



Context

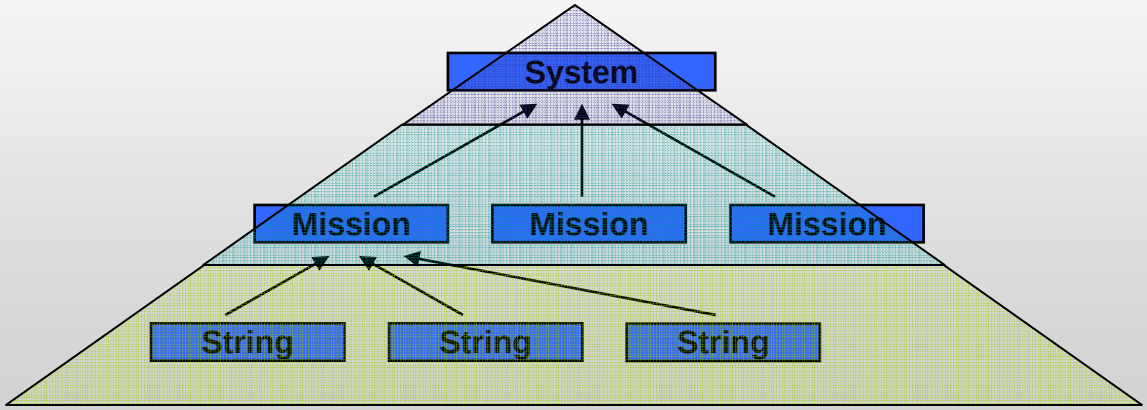
- Node Failure Detection – an “application manager” should place an application on a different node when failure detected.
  - Computing nodes may “disappear” without warning
  - NEED to maintain a base level of service for mission-critical behaviors.
  - If a computation node fails, need to move applications on that node to another node ASAP!!!!
- COTS hardware and software
  - No real-time scheduling.



MLRM Approach to ARMS

Multi-Layered Resource Management: MLRM

- Simultaneously manage multiple QoS concerns.
- Dynamism on multiple levels of abstraction.
  - Infeasible to test all possible dynamic behaviors.
- Adapt to changing resource levels.
  - We want to be able to *certify* the dynamic, multi-layered control system.

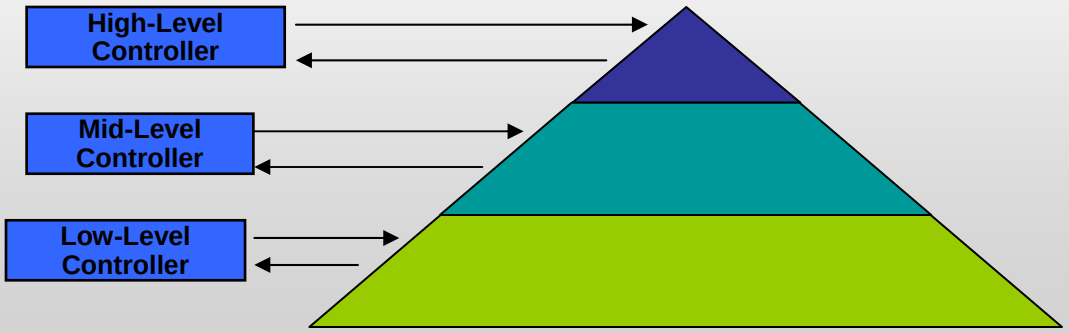


Certification for Dynamic Systems

- Need to be able to evaluate dynamic system behavior in order to perform certification.
  - When system's behavior is insufficient, adjust system behavior.
  - *Feedback control* based on utility.
  - We construct utility functions to measure properties relevant to certification.
- We believe utility driving control is part of evidence for certification.
  - Control driven by utility towards desirable behavior.
  - Certify control as path to certify aspects of system behavior?
- Evidence based.
  - Measure desirable behavior.
  - Control to drive to desirable behavior.

ARMS Program

- We use a ***utility-driven*** approach to measure quality of system's performance.
  - Define utility functions to measure QoS at multiple levels of system hierarchy.
  - Local utility measurements are used as feedback to determine local resource control actions.
    - Utility is a symptom of system quality.
- Want to allow dynamism, but not hinder certifiability.
  - Utility is also a *proxy* measurement of system health.
  - Want to use utility measurements also as artifact for certification.



String Utility

- String Utility is the average utility of its processed jobs.

$$U_i^{S_j} = \frac{1}{P_j} \sum_{l=1}^{P_j} u_l^{job}$$

- Utility of a job is composed of its timeliness and quality factors.
- Job utility assignment is application specific.

$$\sum_{i=1}^{P_j} u_l^{job} = F(T_l^{job}, q_l^{job})$$

- When controlling a string, not vitally important to have in-depth information about other strings.

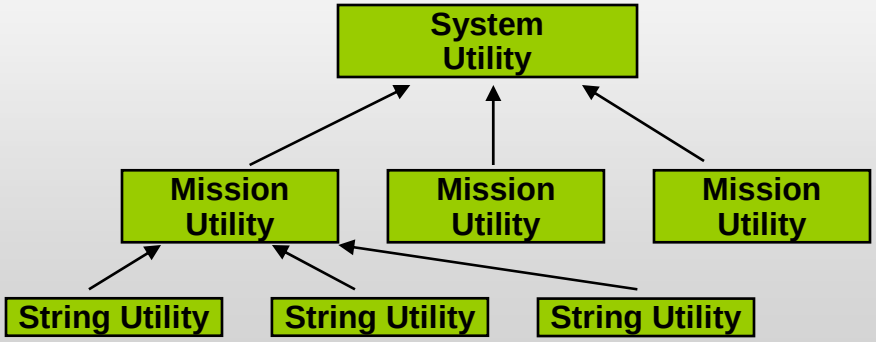
Utility Measurement

- Utility is computed at each level of abstraction.

System utility: $$U = \sum_{i=0}^{M} w_i^m U_i^m$$

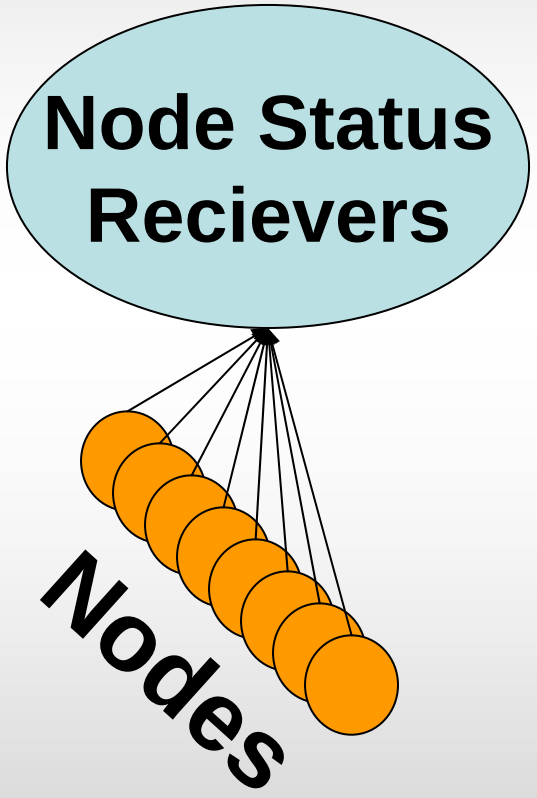
Mission utility: $$U_i^m = \sum_{j=0}^{S_i} w_i^{S_j} U_i^{S_j}$$

String utility: $$U_i^{S_j} = \frac{1}{P_j} \sum_{i=1}^{P_j} u_l^{job}$$



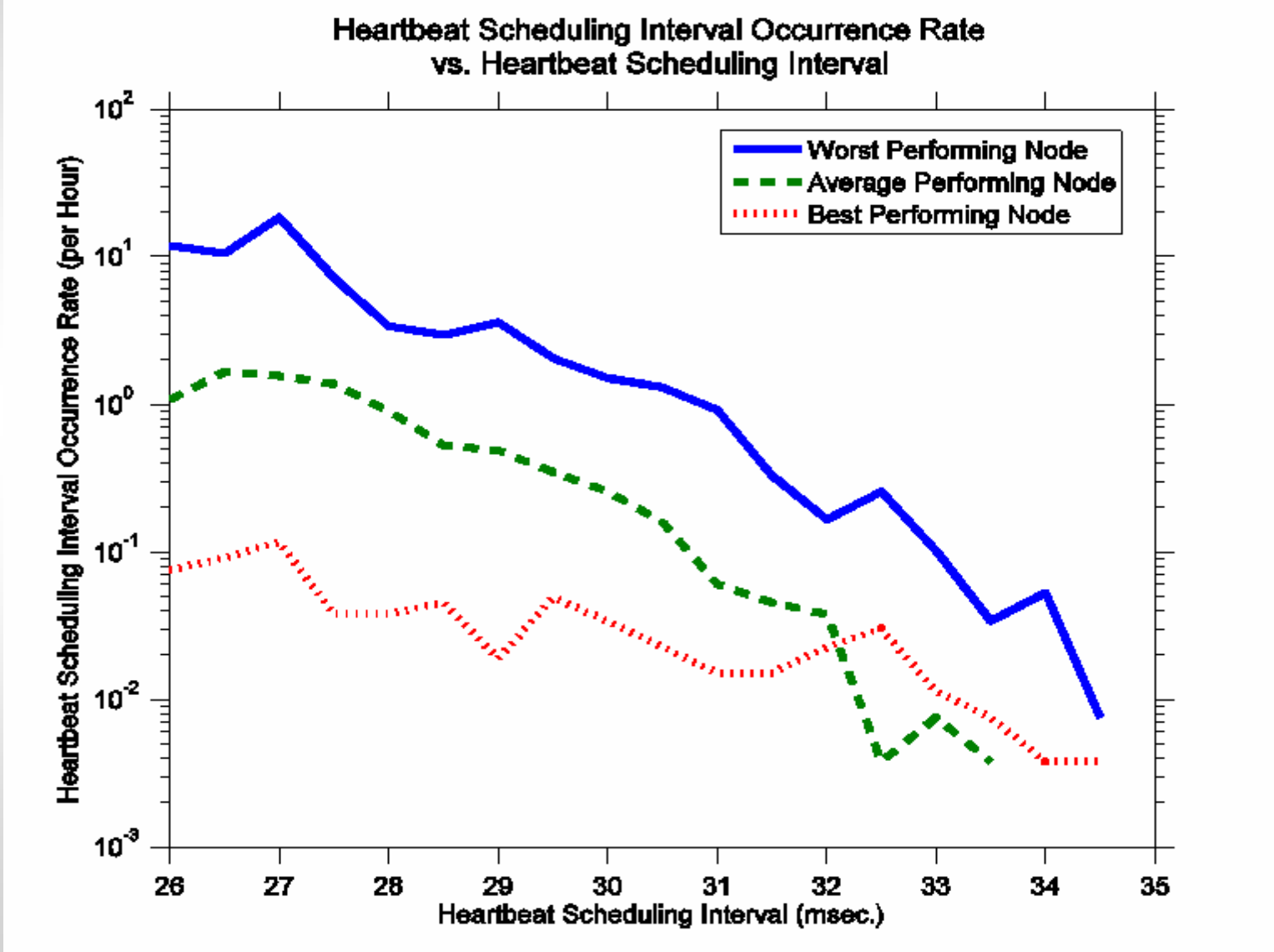
HBFD Case Study:Traditional Approach

- Heart-Beat Failure Detection (HBFD) is a traditional solution to node failure detection.
  - Nodes periodically send “heartbeat” messages to a controller, “Node Status Reciever”
- Drawbacks:
  - Scalability.
  - Due to reliance on real-time processing, inability to handle scheduling errors.
  - Even if have real-time computation, real-time communication is very rare – no way to handle congestion
- A better way is needed that can adapt to system operating conditions and provide “certifiable” real-time, low false-alarm behavior.
  - AI doesn’t cut it for our customer – need guarantees on performance.



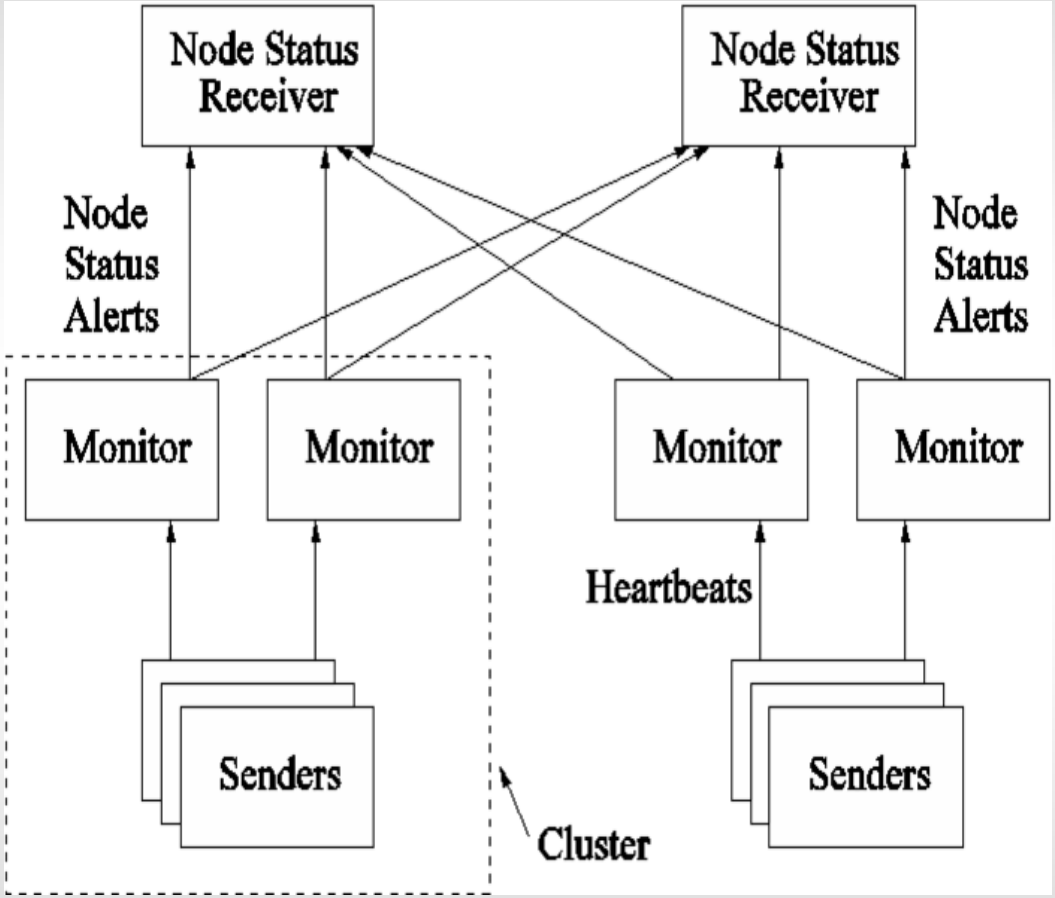
BBN TECHNOLOGIES

Threshold Passing Experiment



A hierarchical architecture

- Each node has a Sender process that periodically sends HeartBeat(HB) messages to 2 local Monitors.
- Monitors perform failure detection operation.
  - This is a non-trivial process for large-scale systems.
  - Control failure-detection threshold to reduce false-alarm rate.
- Failure declarations are sent to two Node Status Receivers (NSR)



- COTS hardware, software, so there are no guarantees on the scheduling of HB transmissions, timely Monitor detection.

NFD String Utility

Want:

$$SM + MN + Th + SI + 3SL < MaxDelay$$

SM is the communication latency of the last heartbeat sent to the Monitor from the failed node. (Can't control.)

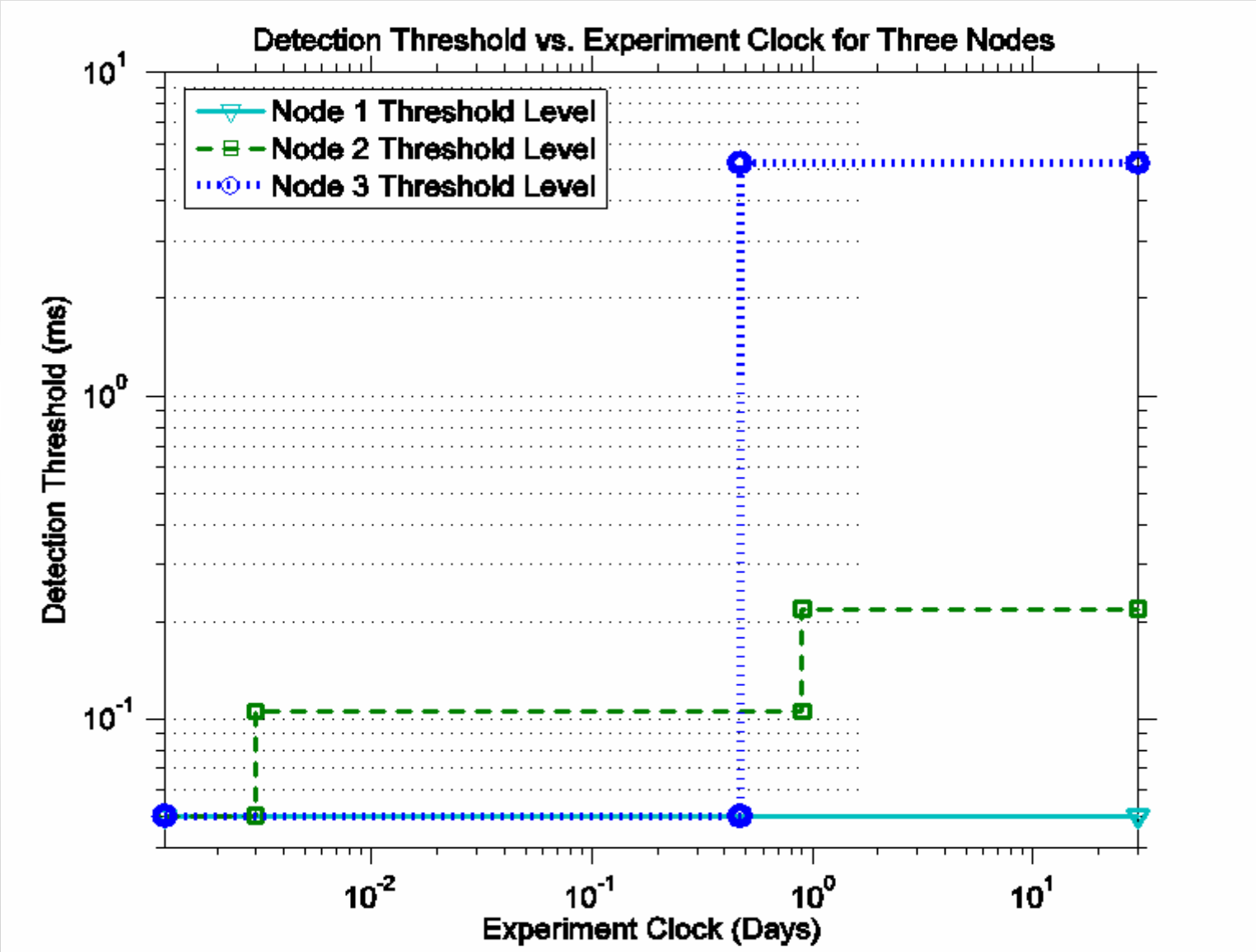
MN is the worst-case communication latency of the failure notification sent from the Monitor to each of the NSR instances. (Can't control.)

Th is the timeout threshold used by the Monitor to detect the node failure. (Can control.)

SI is the period between executions of the sweeper thread in the Monitor. (Can't control.)

SL is the amount of time the sweeper thread takes to run. (Can't control.)

Adaptive NFD Experimentation



Evidence-Based Certification

- Utility measures can capture a large set of attributes of system performance and quality.
  - Utility also measures user-perceived value derived from control system.
  - Utility ultimately provides a quantitative measure for certification.
- Want to use utility measurements as artifact for evidence-based certification.
- Feedback control uses utility measurements to maintain high utility.
  - Allows system to dynamically respond to unforeseen situations.

Ongoing Certification Thoughts

- We should be aware of lifecycle issues!!!!!
  - Certification is only valid until we modify a system.
- Of course, always need to certify new components.
  - Still need to be wary of how new components might interact with established infrastructure.
- Can we exploit encapsulation effects due to feedback to aid in rapid recertification as system components are used in new contexts?

Research Roadmap - 5 to 10 years

- Develop techniques to enable composition of certifiable components for a certifiable system:
  - Identify clear interfaces for the interactions of components, not just functional, but also QoS (through system resources, time, etc.).
  - Clearly identify good dynamic behavior from bad dynamic behavior in the composed system.
  - Produce controls that ensure good dynamic behavior and prevent bad dynamic behavior.
- A path toward this vision:
  - (Short term) Use clear partitioning mechanisms, such as resource reservations.
  - (Medium term) Use priority based controls, to provide needed bounding. Requires research in analysis, interfaces, design, and runtime feedback and enforcement.
  - (Longer term) Removing constraint mechanism interfaces in favor of policy-driven application control.
    - Want to automatically regulate component interaction.
    - Requires research in sequential process languages, specification of certification behaviors, composition of specifications, and policy driven control.

BBN TECHNOLOGIES

Thank You!

Kurt Rohloff

BBN Technologies

krohloff@bbn.com