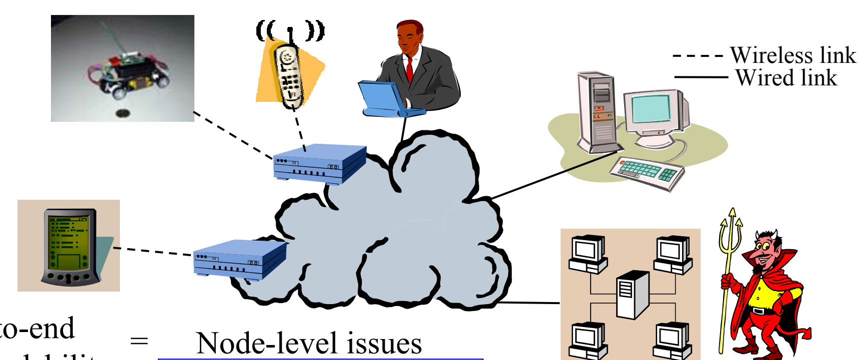
Non Intrusive Detection & Diagnosis of Failures in High Throughput Distributed Applications

Saurabh Bagchi
Dependable Computing Systems Lab (DCSL) &
The Center for Education and Research in Information Assurance and Security (CERIAS)
School of Electrical and Computer Engineering
Purdue University

Joint work with: Gunjan Khanna, Ignacio Laguna, Fahad Arshad (Students); Gautam Kar (IBM); Peter Shier (Microsoft)

Work supported by:
IBM, Purdue Research Foundation

DCSL Research Thrusts



End-to-end Dependability = Node-level issues

Applications
Middleware
Reliable communications
Operating system
Hardware

\+ Network-level issues

Research Projects in DCSL

- Framework for distributed intrusion tolerant system
  - How to build an adaptive infrastructure for diagnosing and recovering from failures in a distributed platform?
  - Application: Distributed e-commerce application, Voice over IP system
- Black-box diagnosis
  - How to diagnose source of errors in high throughput distributed apps?
  - Application: Distributed e-learning application, Distributed e-commerce application
- Dependable ad-hoc and sensor networks
  - How to build dependable network out of inherently unreliable components with resource constraints?
  - Application: Monitoring environmental conditions in urban areas

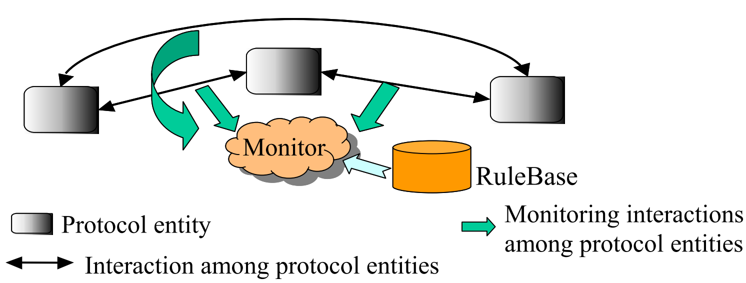
Monitor Project: Motivation

- Distributed Network Protocols are used in everyday computing
- Increased reliance on these protocols for critical applications
  - Financial Sector, Telecommunications, Security etc.
  - Cost of downtimes of these systems can run into several millions of dollars
- Vulnerable to naturally occurring errors and malicious attacks
  - Response failure, Timing failure, Human mis-configurations

Design Goals

- A generic framework which should address *both* detection and diagnosis
  - Detection: Evidence of a protocol behavior which differs from the defined set of correct actions
  - Diagnosis: Process to pin-point the root cause of the detected failure
- Fault tolerance system is non-intrusive to the application
  - Application is treated as black-box
  - No explicit probes during runtime
  - Two systems operate asynchronously
- Fast runtime detection and diagnosis
  - Recovery is made more feasible

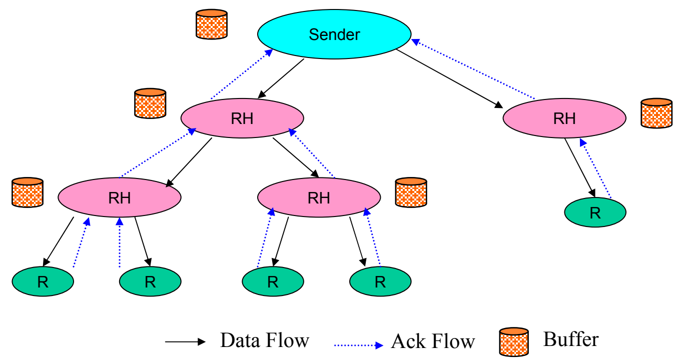
Solution Approach: Monitor



- Monitor provides the fault tolerance services
- It overhears message exchanges between **P**rotocol **E**ntities (PEs)
- For detection, Monitor matches message interactions against an anomaly-based rule base
- For diagnosis, Monitor creates a dependency graph and runs a rulebase over the deduced state variables of the PEs

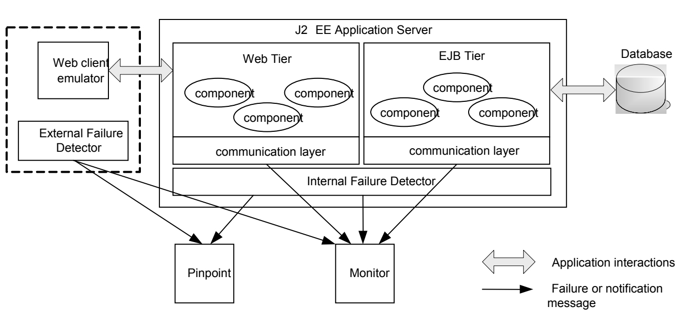
Application Example

- Distributed e-learning based application at Purdue
  - Uses tree based reliable multicast (TRAM)
  - Reliable delivery of multimedia stream to a large number of receivers
  - Ensures reliability of message transfer in case of node or link failures and message errors.

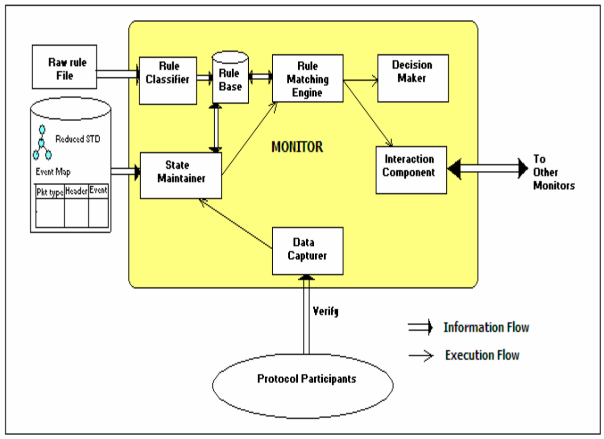


Application Example

- Distributed e-commerce application based on J2EE middleware
  - PetStore: Buys and sells pet related supplies with additional system administrator and supplier interactions
  - Three tier e-commerce system



Monitor Architecture



1. **Data Capturer:**
   - Snoops over communication between PEs.
2. **State Maintainer:**
   - Contains event definitions & reduced STDs.
   - Flags rule matching based on State×Event
3. **Rule Classifier:**
   - Decides if rules are to be matched at current monitor.
4. **Interaction Component:**
   - Responsible for interactions between Monitors for distributed rule matching.

Structure of Rule Base

- Rule matching engine invoked by State Maintainer
- Rules defined based on protocol specifications *and* QoS requirements.
- Rules are anomaly based
- Currently created manually by sysadmin or from UML specifications
- Rules can be
  - Combinatorial: Valid for entire duration except for transients
  - Consists of expressions of state variables arranged as an expression tree yielding Boolean result
  - Temporal: Associated time component for precondition and postcondition

Detection: Challenges & Solutions

- How to overhear the message interactions between PEs
  - Passive approach: Broadcast communication medium; Port mirroring on router
  - Active approach: The middleware on which PE runs is modified to forward messages to Monitor
- Efficient rule matching
  - Customized syntax for rules is developed based on Temporal Logic Actions (TLA)
  - Example: $L \leq |V_t| \leq U,\ t \in (t_i, t_{i+k})$
  - Highly efficient rule matching for each incoming event for each type of rule
- Scalability
  - Hierarchy of local and non-local monitors
  - Filtering at each level
  - Most interactions are local and hence configuration is optimized

Sample Failure Scenarios

- E-learning application
  - Receiver is faulty; Withholds Ack causing slow data rate and high buffer occupancy; Example of error propagation across several PEs
  - A repair head runs out of buffer space; Unable to handle the requisite number of receivers
- E-commerce application
  - Database lock is not released; An update transaction by the supplier application is blocked indefinitely
  - A malicious user accesses private data from the back-end database

Results: E-learning Application

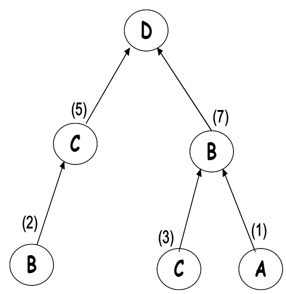
- MPEG-2 video stream with single server, multiple clients
- Minimum data rate – 20 KB/sec, Max data rate – 40 KB/sec
- Errors injected in bursts – burst length = 15 ms.
- Error models: Stuck-at-Fault; Directed; Random
- Loose clients check data rate after 4 Ack windows, tight clients after every Ack window.
- Possible outcomes – Exception (E), Client crash (C), Data rate error (DE), No failures (NF)
- Single level Monitor has accuracy 84.37%
- Hierarchical Monitor has accuracy of 90.97%, 7% more than in the single Monitor case

Diagnosis: Challenges

- Error propagation could cause multiple simultaneous failures
  - Fast error propagation makes diagnosis difficult since chain may span multiple PEs
  - Existing systems consider entity manifesting failure to be faulty
- Under failure condition, further stress in the system is to be avoided
  - Monitor should not send active probes/tests to the application system for diagnosis
- There are various factors causing uncertainty
  - Messages may be lost
  - Monitor capacity may temporarily get overwhelmed
  - PEs may be able to mask some errors

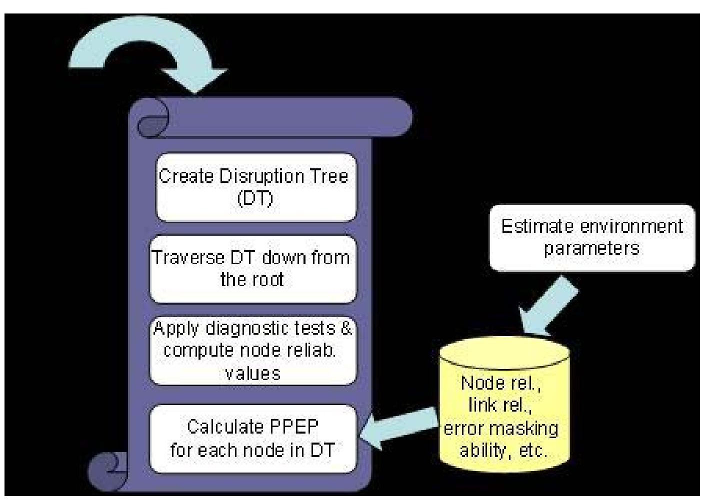
Diagnosis Approach

- Monitor maintains a *causal graph* with events ordered according to the logical time
- On detection of a violation at n_f, diagnosis starts by building a suspicion tree of all the nodes which sent messages to n_f say set A
- Each node in the suspicion set is tested using a test procedure
- If no node is faulty, then suspicion set is expanded to include the nodes which sent messages to nodes in A



- The path P from any node n_i to the root n_f constitutes a possible path for error propagation
- PPEP(n_i, n_f) is defined as the probability of node n_i being faulty and causing this error to propagate on the path from n_i to n_f, leading to a failure at D

Diagnosis Flow-Chart



Sample Failure Scenarios: E-commerce Application

- Null call: Instead of returning the appropriate value, a method returns a null object
- Consequently, the EJB does a dummy operation and the database is not updated
- A subsequent transaction sees an error because it is reading an incorrect database state

Sample Result

- Achieves high accuracy as does the best of breed static dependency graph scheme (Pinpoint)
- But incurs much fewer false positives
- Performance dependent on length of chain, resources at Monitor, quality of rules

What's in the works

- High throughput applications
  - More efficient per packet processing
  - Sampling so that some of the packets are not processed and discarded
  - Challenges: Maintain accuracy, decide which packets to drop
- Handling failures in the Monitor
  - Replication to tolerate environment-specific errors
  - Leverage work on synthetically introducing diversity to create diverse replicas
  - Challenges: Low overhead replica coordination

Conclusion

- We have a system (the Monitor) that can do low overhead detection and diagnosis in black-box distributed applications
- Applied to four real applications so far
  - Distributed e-learning
  - Distributed e-commerce
  - Virtualized server environment (IBM)
  - Device drivers in Windows XP (Microsoft)

Publications

1. Gunjan Khanna, Saurabh Bagchi, Kirk Beaty, Andrew Kochut, and Gautam Kar, "Providing Automated Detection of Problems in Virtualized Servers using Monitor framework," In the Workshop on Applied Software Reliability (WASR), held with the IEEE International Conference on Dependable Systems and Networks (DSN), 6 pages, June 25-28, 2006.
2. Gunjan Khanna, Padma Varadharajan, and Saurabh Bagchi, "Automated Online Monitoring of Distributed Applications through External Monitors," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 2, pp. 115-129, Apr-Jun, 2006.
3. Nipoon Malhotra, Shrish Ranjan, and Saurabh Bagchi, "LRRM: A Randomized Reliable Multicast Protocol for Optimizing Recovery Latency and Buffer Utilization," In the 24th IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 215-225, October 26-28, 2005, Orlando, Florida.
4. Gunjan Khanna, Padma Varadharajan, and Saurabh Bagchi, "Self Checking Network Protocols: A Monitor Based Approach," In Proceedings of the 23rd IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 18-30, October 18-20, 2004, Florianopolis, Brazil.

DCSL URL: www.ece.purdue.edu/~dcsl