NON INTRUSIVE DETECTION AND DIAGNOSIS OF FAILURES IN HIGH THROUGHPUT DISTRIBUTED APPLICATIONS

Gunjan Khanna

PhD Final Examination

Advisor : Prof. Saurabh Bagchi
Committee Members: Profs. Ness Shroff, Cristina Nita-Rotaru, and Rudolph Eigenmann

Dependable Computing Systems Lab
ECE, Purdue University

Research Initiatives

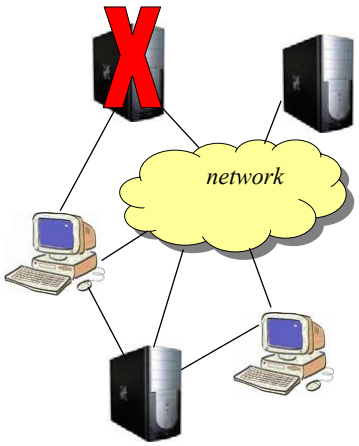
- Detection & Diagnosis in Distributed Systems
  - Hierarchical Monitor Framework: Generic, re-configurable, non-intrusive, and scalable
  - PRDC'04, SRDS'04, WASR'06, DSN'06 (fast abstract), TDSC'06, ICSOC'06, TDSC'07, SRDS'07 (2-submissions)
- Virtual Machine Management
  - Providing management solutions centered around fault tolerance semantics to the domain of virtualized server scenarios (jointly with IBM Research)
  - NOMS'06, WASR'06
  - 2 patents have been filed
- Fault Tolerant Data Dissemination in Sensor Networks
  - Provide push-pull based primitive for reliable communication: SPMS, SPMS-Rec
  - DSN'03(fast abstract), DSN'04, WCNC'07, VTC'07

Outline of the Talk

- Background
  - Background on detection and diagnosis: Summarize detection and probabilistic diagnosis approach
- Tasks from Prelim Examination
  - Reduction of State Space
  - Comparison with Pinpoint
  - Scaling the Monitor approach: Sampling
- Sampling Technique
  - Motivation
  - Solution Approach
  - Experiments and Results
- Related Research
  - High Throughput Detection
- Conclusions and Future Work

Motivation

- Distributed network protocols are integral in all sectors
  - File servers, databases, e-commerce applications, p2p etc.
- Increased reliance on these protocols for critical applications
  - Financial, Telecommunications, Security etc.
  - Cost of downtimes of these systems can run into several millions of dollars
    - Financial broker $6.2M/hr (Source: International Data Corporation, 2005)
- Lack of a comprehensive detection and diagnosis framework



Detection and Diagnosis is imperative to improve reliability

Design Goals: Detection and Diagnosis

- ***Detection***: Evidence of a protocol behavior which differs from the defined set of correct behavior
- ***Diagnosis***: To be able to pin-point the root cause of the failure
- A generalizable framework which should provide Detection and Diagnosis
- Treat application entities (or protocol entities) as Black-box
  - Non-intrusive approach
  - Operate asynchronously
- Online mechanism enforcing low latency and high accuracy
- Autonomic in nature requiring minimum expertise to operate

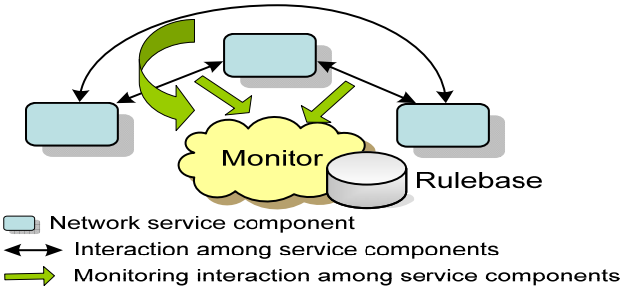
Hierarchical Monitor: Real Time Solution

Detection System

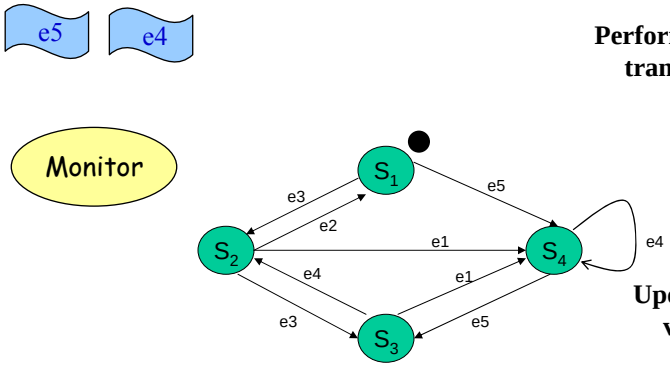
- Logical separation between the protocol entities (PEs) from monitoring entities
- Define categories of rules for matching (Anomaly based)
- Fast matching

Diagnosis System

- Causal dependencies are tracked
- Monitor deduces the protocol state
- Non-intrusive diagnostic tests are used
- Black-box diagnosis
- Distributed in nature
- Probabilistic modeling with rich set of parameters



Use of State Transition Diagram



Perform state transition

Update state variables

Instantiate Rules

Example State Transition Diagram (STD)

Road Map

- Background
  - Brief about Detection and Diagnosis: Summarize detection and probabilistic diagnosis approach
- Leftover from Prelim Examination
  - Reduction of State Space
  - Comparison with Pinpoint
  - Scaling the Monitor framework
- Sampling Technique
  - Motivation
  - Solution Approach
  - Experiments and Results
- Related Research
  - High Throughput Detection
- Conclusions and Future Work

Prelim Presentation Slide: Future Work

1. Autonomic STD Reduction
   - Larger STD causes more links in the causal graph and hence increases the size of diagnosis tree
   - All states and events might not be visible, or might not have rules associated with them
   - Reduce internal states and states which do not have rules: Makes the Monitor architecture more flexible
2. Comparison with other approaches and generalization
   - Applications are composed of multiple services interacting through messages
   - Currently working on testing the approach on a e-commerce test-bed using PetStore
3. Detection and Diagnosis in high rate network streams
   - We would like to push the *knee* to the right
   - Provide intelligent sampling so as to keep the missed alarms and false alarms low
   - Would the detection or diagnosis model require a change ?
   - $\|\mathbf{R}_{sv} - \mathbf{R}_s\| < \epsilon.\mathbf{R}$

1. Autonomic STD Reduction

- Provide offline mechanisms to reduce the states
- Two state reduction mechanisms are proposed: Invisible states and rule-less states
- Prove transparency of the detection and diagnosis process to the new state reduction process: Monitor should provide the same detection and diagnosis results on the original and reduced STD
- Rigorously test the effectiveness through experiments on reliable multicast protocol TRAM and large random STDs.
- On an average provide over 40% reduction in latency
- Co-contributor: Mike Yu Cheng

2. Monitor Comparison with Pinpoint

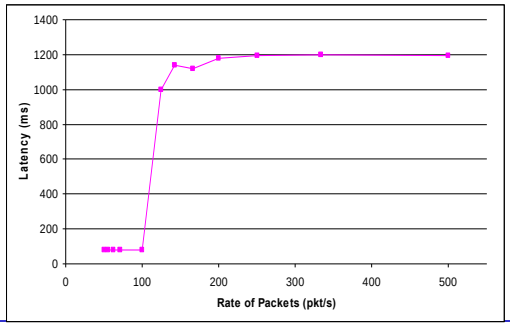
- Pinpoint, an offline diagnosis approach developed by Chen *et al.* 2002, to address problem determination in internet services
  - Pinpoint clusters the components touched by failed transactions to deduce the most likely cause of failure
- We implement the Pinpoint algorithm and test both approaches on 3-tier e-commerce test bed
  - PetStore application on JBoss application server
  - 55 client transactions and 4 different type of fault injections
- Monitor outperforms Pinpoint
  - Monitor and Pinpoint achieve same accuracy but Monitor has higher precision
  - Monitor has much lower false positives compared to Pinpoint
- Co-contributors: Ignacio Laguna, Fahad Arshad

Road Map

- Background
  - Brief about Detection and Diagnosis: Summarize detection and probabilistic diagnosis approach
- Leftover from Prelim Examination
  - Reduction of State Space
  - Comparison with Pinpoint
  - Scaling the Monitor approach
- Sampling Technique
  - Motivation
  - Solution Approach
  - Experiments and Results
- Related Research
  - High Throughput Detection
- Conclusions and Future Work

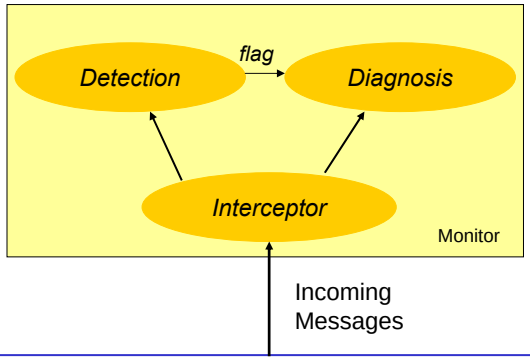
Scalability: Motivation

- Monitor breaks beyond a particular rate of incoming packets (or adding more protocol entities for verification)
  - Increase in the latency of detection
  - Loss of accuracy
- Monitor should be applicable to high rate data streams
  - Should be able to verify a large number of protocol entities



Scalability Challenges

- Computational and memory constraints
- Stateful approach requiring state transitions
- Rule matching is performed for messages which might be temporally distant

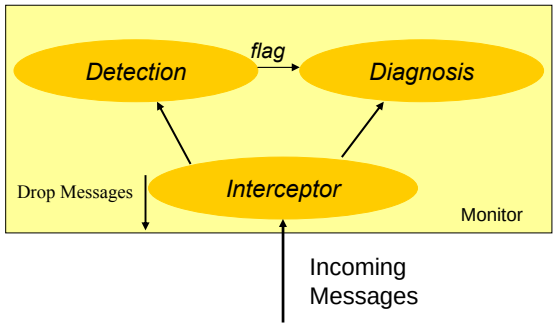


Scalability: Design Goals

- Latency and accuracy should not drop drastically
  - Graceful degradation of latency and accuracy
- Monitor should be executable on off-the-shelf hardware
  - Should not have large memory footprint
  - Reduce computations
- Stateful approach should be followed
  - Natural errors in systems are stateful
  - Example: Failures in Windows NT
  - Example: Failure prediction in cycle-sharing systems

Road to Developing a Solution (1)

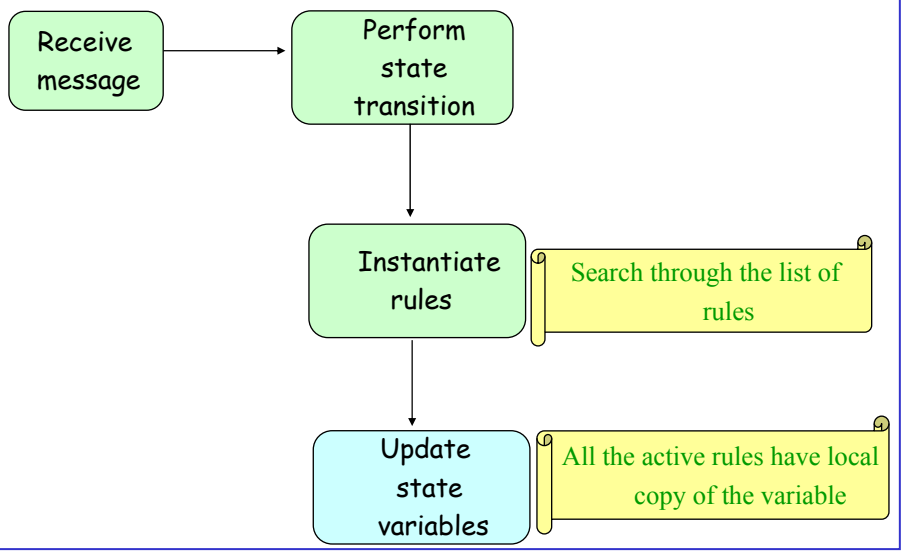
- Detection workload can be represented as
  - *Work per unit time = rate of incoming messages × the amount of work performed for each message*
- Minimize the cost of processing for each message
  - Better data structures
- Sample the incoming messages which the Monitor has to process



Existing Rule Matching

- Rules defined based on protocol specifications *and* QoS requirements
- Rules are anomaly based
  - Define the correct behavior of the protocol
- Five generic temporal rule categories
  - Example:
    - The Hello message count should be between 10 and 30 for the next 5000 msec. (*QoS*)
    - Sender should receive an Ack after sending 32 Data packets (*protocol specification*)

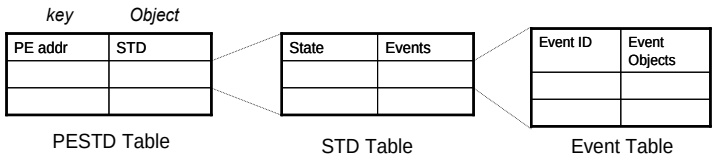
Road to Developing a Solution (2)



Efficient Rule Matching – *Monitor-HT* (1)

- Rationale
  - Provide efficient look-up using hashtables
  - Eliminate duplicate copies of the state variable

- State Transition Diagram is organized in a multi-level hashtable
  - Constant Order look-up



Efficient Rule Matching – *Monitor-HT* (2)

Data

Rule 1	[illegible]
Rule 2	[illegible]
Rule 3	[illegible]

Previous Approach

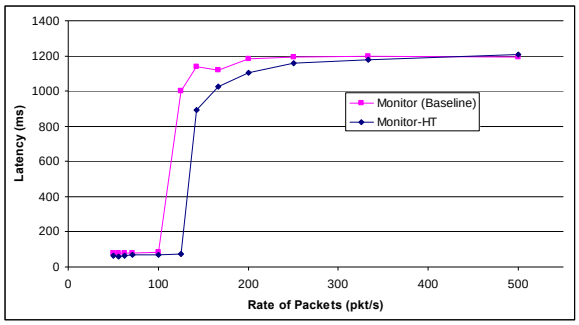
Data

State Var	[illegible]
Rule 1	1
Rule 2	0
Rule 3	1

New Approach

- Multiple rules are matching the same message type
  - Local variables contain snapshots of the global count at instantiation and at matching instant
  - *PE* × *Event ID* tuple is only incremented once

Monitor-HT versus Monitor-Baseline



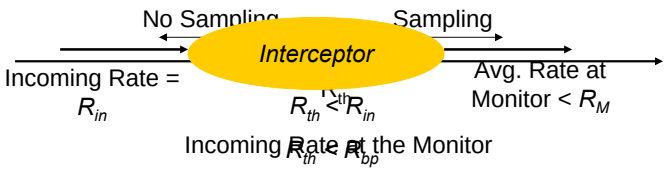
- We compare the latency of detection of Monitor-Baseline and Monitor-HT on a reliable multicast protocol TRAM
- Latency is measured from instantiation of rule to the end of rule matching
- Monitor-HT achieves a 25% higher breaking point in terms of rate of incoming packets

Our Solution Approach: Sampling

- Monitor-HT still has to perform a minimum constant amount of work for every incoming message
  - Modify Monitor-HT to reduce the incoming workload
- Instead of processing every message, ***sample*** the incoming messages (*Monitor-S*)
  1. How and what sampling approach should be taken?
  2. How are the rules modified due to sampling?
  3. How does Monitor-S track the PE's STD in the presence of sampling?
- Uniform random sampling
  - Uniform random method is oblivious to the incoming message type
  - Any sampling approach based on the information of the incoming message will require some processing of the message before sampling
- We choose uniform random sampling: rate of sampling is dependent on the rate of incoming messages

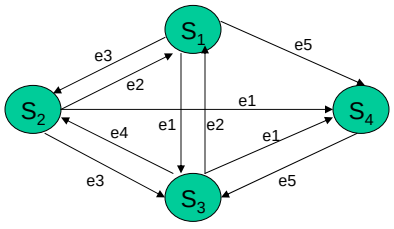
How are the rules modified ?

- Assume Monitor-Baseline achieves a desired latency and accuracy uptil R_{bp} rate of incoming messages
  - Choose $R_{th} < R_{bp}$
- If the incoming rate $R_{in} > R_{th}$
  - drop message at the rate of 1 in every $R_{in} /(R_{in} - R_{th})$ messages
  - Incoming rate is recalculated after a window of 30 seconds
- Rules are designed by the system administrator for actual application system and not the sampled stream seen by Monitor-S
- Scale the constants in the rules by a factor of R_{th} / R_{in}
  - "*receive 10 **Acks** in 100 sec*" then because of sampling the rule is modified to "*receive 10.(R_{th} / R_{in}) **Acks** in 100 sec*"



How does Monitor-S track the PE's STD?

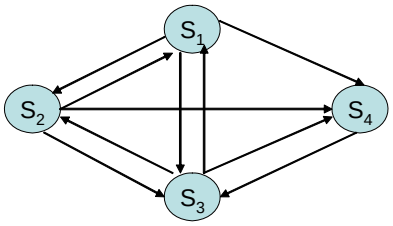
- Monitor framework keeps track of the state of the entity for performing detection
- Dropping a message can cause Monitor-S to lose track of the current state of the entity



Example State Transition Diagram (STD)

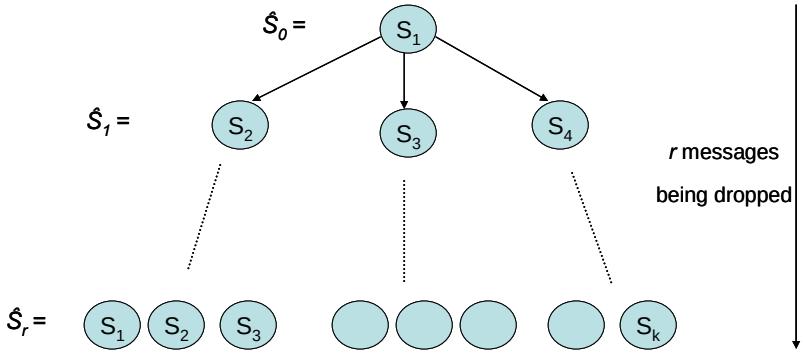
State Vector ($\hat{S}$)

- Instead of keeping a single current state for the application protocol entity, keep a vector of possible states
  - $\hat{S} = \{S_1, S_2 \ldots S_K\}$
- If r consecutive messages are dropped starting from state S_{start} then the state vector $\hat{S}$ consists of the union of states reachable in r steps from S_{start}
- Computing the state vector at runtime: Expensive !
- Compute state vectors Offline



Starting with state S_1 if 1 message is dropped, then state vector is given by:
$\hat{S}_1 = \{S_2, S_3, S_4\}$

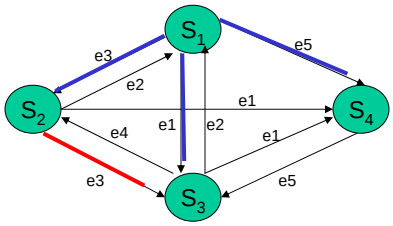
Example



Example State Vectors at a each level. A depth j represents the state vector if j messages are dropped consecutively from S_1

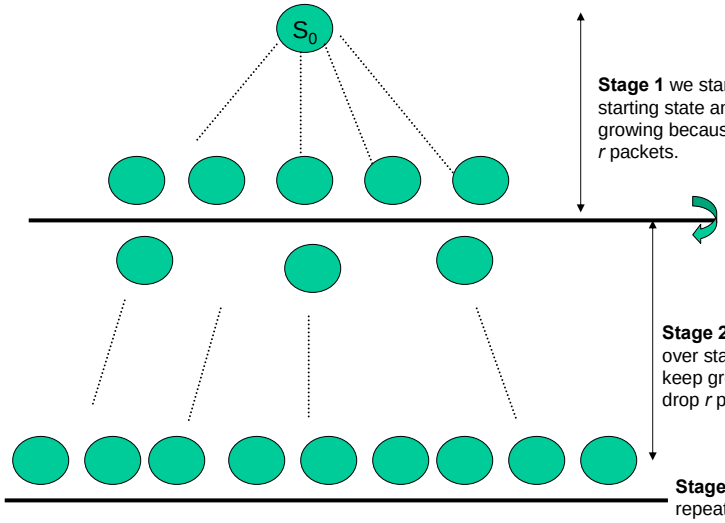
Reduction in size of State Vector

- Size of the state vector does not keep growing
  - Bounded by the total number of states
  - Sampling of a message
- Sampling a particular message causes the size to reduce
- Example: Consider the STD below
  - At start: $\hat{S} = \{S_1\}$
  - Drop a message: $\hat{S} = \{S_2, S_3, S_4\}$
  - Sample a message (say e3): $\hat{S} = \{S_2\}$



Example State Transition Diagram (STD)

Stages of Sampling Approach



Stage 1 we start with one starting state and keep growing because we drop *r* packets.

At this boundary we sample a packet and reduce the states to some number j

Stage 2 we start with left over states and again keep growing because we drop *r* packets.

Stage 3 the system repeats itself in the same manner as above…….

Analytical Bounds

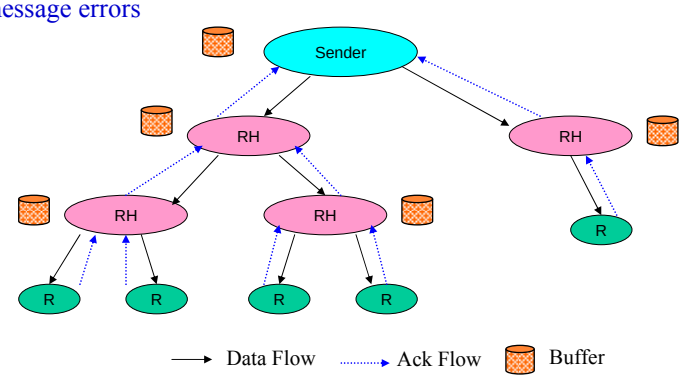
- Memory requirement is large to store the pre-computed state vectors
  - For a *k*-regular graph, it is $k(k^r-1)/k-1$ if *r* consecutive messages are dropped
  - Use a bit vector representation: proportional to $S^2.r$ bits *;* where S is the total state size
- Size of the state vector determines the number of rule instantiations and hence the overall computation
- For a *k*-regular graph (representing the STD), we show that the size of the state vector is asymptotically bounded if

$$r = min(log_k M - 1, log_k z)$$

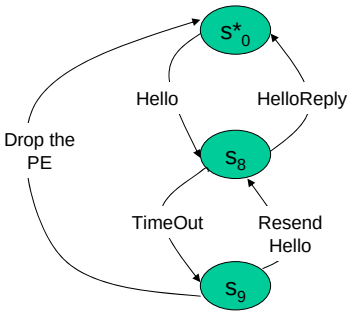
  - *r* is the number of consecutive packets which can be dropped
  - z is the number of different types of messages present in the STD
  - *M* is the total number of outgoing links for all the states in the state vector $\hat{S}$

Example Protocol : TRAM

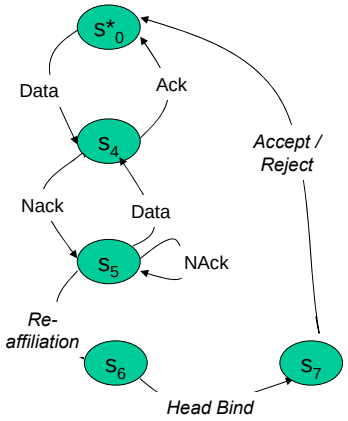
- Tree Based Reliable Multicast Protocol (TRAM)
  - It is a scalable protocol aimed to function in large area networks with hundreds of participants
  - Ensures reliability of message transfer in case of node or link failures and message errors



Example State Transition Diagrams (STD)



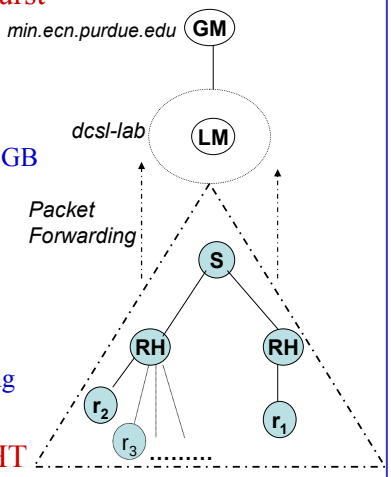
Liveness messages



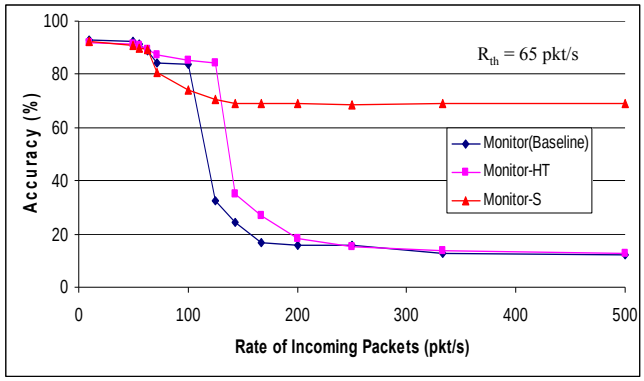
Data-Ack messages and re-affiliation

Experimental Set-Up

- TRAM is used as the application protocol and fault injection is performed for a burst length
- Monitor and TRAM run on separate machines
  - Desktop PCs with 2.4GHz processor and 1GB RAM
- We measure the *accuracy* and *latency*
  - Accuracy is (1-missed alarms)
  - Latency is measured from start of rule instantiation to the time it took for matching
- Compare Monitor-Baseline, Monitor-HT and Monitor-S

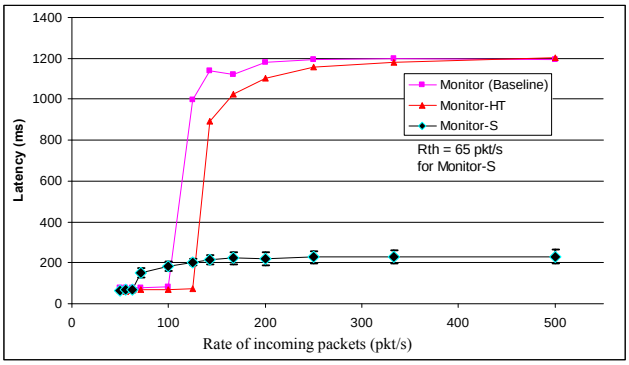


Accuracy Results (Sender-Receiver)



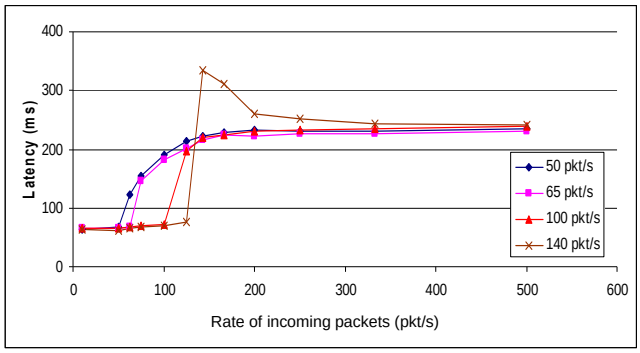
- Monitor-Baseline and Monitor-HT break at 100 pkt/s and 125 pkt/s respectively
- Monitor-S has a small decrease in accuracy but it still maintains accuracy at ~ 70% compared to Monitor-HT's 16% accuracy

Latency Results (Sender-Receiver)



- Similar to accuracy results Monitor-S has a marginal increase in the latency with increasing packet rate as compared to Monitor-HT and Monitor-Baseline which have a collapse
- Monitor-S provides detection at a low latency of ~200ms as compared to 1200ms for Monitor-Baseline for high data rates

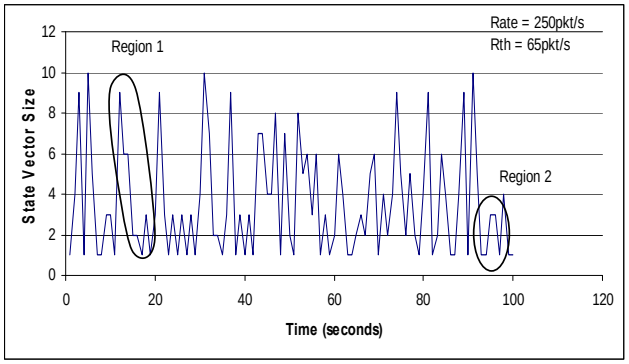
Effects of Varying R_{th} : Latency



- Here for the plot of R_{th} = 140 pkt/s we see a sharp spike because Monitor-HT's breaking point(R_{bp}) is at 125 pkt/s
- R_{th} should be appropriately chosen, preferably far below R_{bp} to account for inaccuracies in estimating R_{bp} and fast fluctuations in incoming data rate

PURDUE UNIVERSITY

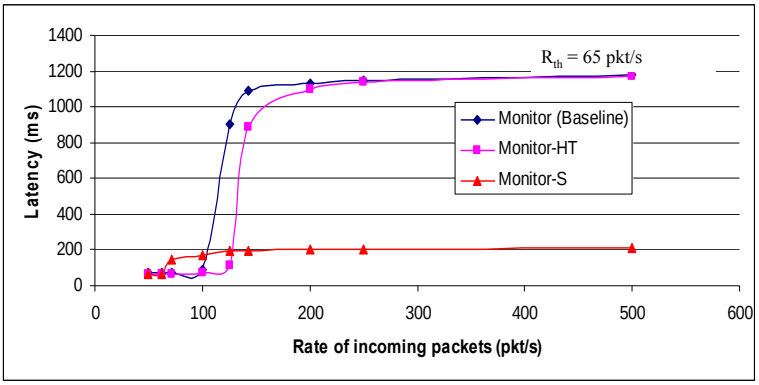
Variation in Size of State Vector (|Ŝ|)



- Sample run of Monitor-S measuring |Ŝ| at receipt of every alternate packet
- In Region 1, |Ŝ| drops in steps from 9 to 6 and finally to 1. The drop in |Ŝ| is because of the unique possibility of the sampled event in only some of the states
- In Region 2, |Ŝ| increases from 1 to 3 because of a message drop

PURDUE UNIVERSITY

Latency Results (Sender-RH-Receiver)



- We repeat the experiments with sender-RH-receiver topology
- Single sender, 2 RHs and 2 receivers, one receiver under each RH
- We observe similar results as in sender-receiver scenarios

Road Map

- Background
  - Brief about Detection and Diagnosis: Summarize detection and probabilistic diagnosis approach
- Leftover from Prelim Examination
  - Reduction of State Space
  - Comparison with Pinpoint
  - Scaling the Monitor approach
- Sampling Technique
  - Motivation
  - Solution Approach
  - Experiments and Results
- Related Research
  - High Throughput Detection
- Conclusions and Future Work

Related Research

- Change Detection in Networking
  - Sketch based approaches: Deltoids, Infocom'05, Infocom'06
  - Develop statistical models to describe the stream behavior
  - In Monitor state of the application is closely examined and it accounts for spikes as well. Provides flexibility to switch to sampling or no-sampling
- Stateful Detection
  - Particular attention from the security community in building Intrusion Detection Systems
  - Snort uses aggregated information from TCP packets to make decisions
  - SciDive provides stateful detection engine for VoIP
  - Restricted to the domain and focussed on accuracy
- Detection in Distributed Systems
  - Heartbeats, watchdogs
  - Detection of Failures using event graphs

Contributions of the Research Initiative

- We proposed a generic hierarchical framework black-box system – the Monitor, to provide non-intrusive detection and diagnosis in distributed systems
- We developed a stateful detection mechanism that can scale to a high data rate of the application protocol
- The Monitor can account for uncertainties of the deployment environment as well as imperfect knowledge of the characteristics of the protocol entities
- We provide state reduction methods to address the problem of state space explosion
- Scalability is achieved by a sampling approach which reduced the overall workload at the Monitor for a given message rate

Future Work

- Autonomic Recovery
  - Recovery is the logical next step after detection and diagnosis of failure
  - How to provide autonomic recovery in the current framework
  - Fault Tolerant Community lacks model for generic autonomic recovery: More work is needed to fully understand the potential of autonomic recovery
- Application of the Monitor framework in other scenarios
  - System Management in Virtualized Server Environments
    - Virtual machines are emerging as a new paradigm for distributed computing
    - Virtualization, in its microcosm, brings a whole new challenge to system management. The increased layer causes increased complexity and makes it harder for a system administrator to find and resolve failures
  - Windows Device Drivers
- Modeling of the Monitor Framework
  - Develop more accurate theoretical models

Publications: Monitor Project

- **Journal**
  - "**Automated Rule-Based Diagnosis in Distributed Systems,**" G. Khanna**,** P. Varadharajan, Y. Cheng, S. Bagchi, M. Correia, and P. Verissimo, accepted in IEEE Transactions on Dependable and Secure Systems (TDSC), May 2007.
  - "**Automated Online Monitoring of Distributed Applications Through External Monitors,**" G. Khanna**,** P. Varadharajan, and S. Bagchi, in IEEE Transactions on Dependable and Secure Computing (TDSC), Feb. 2006.
- **Conference and Workshops**
  - "**Stateful Detection in High Throughput Distributed Systems,**" G. Khanna, I. Laguna, F. Arshad, and S. Bagchi, *in submission* to SRDS 2007.
  - **"Probabilistic Diagnosis through Non-Intrusive Monitoring in Distributed Applications,"** G. Khanna**,** I. Laguna, F. Arshad, and S. Bagchi, *in submission* to SRDS, 2007.
  - **"State Space Reduction for efficient Detection and Diagnosis in Distributed Systems,"** G. Khanna, Y. Cheng, S. Bagchi, *in submission* 2007.
  - **"Self Checking Protocols: A Step Towards Fault Tolerance in Services"** G. Khanna, in ICSOC, PhD Symposium, 2006.
  - **"Providing Automated Detection of Problems in Virtualized Servers using Monitor Framework,"** G. Khanna, S. Bagchi, K. Beaty, A. Kochut, N. Bobroff, and G. Kar, in Workshop on Applier Software Reliability (WASR) held in conjunction with DSN, 2006.
  - "**Modeling Probabilistic Diagnosis Parameters**," G. Khanna, Y. Cheng, and S. Bagchi, Fast Abstract in Dependable Systems and Networks (DSN), 2006.
  - **"Self Checking Network Protocol: Monitor Based Approach,"** G. Khanna, P. Varadarajan, and S. Bagchi, In Symposium on Reliable and Distributed Systems, (SRDS), pp. 18-30, Florianopolis, Brazil, 2004.
  - "**Failure Handling in a Reliable Multicast Protocol for Improving Buffer Utilization and Accommodating Heterogeneous Receivers,"** G. Khanna, J. S. Rogers, and S. Bagchi, Pacific Rim Dependable Computing (PRDC), 2004.

Other Publications

- **Conference and Workshops**
  - **"Performance comparison of SPIN based Push-Pull Protocols"** R. Khosla, X. Zhong, G. Khanna, S. Bagchi, and E. J. Coyle, in Wireless Communications and Networking Conference (WCNC), 2007.
  - **"Data Centric Routing in Sensor Networks: Single-hop broadcast or Multi-hop unicast?,"** R. Khosla, X. Zhong, G. Khanna, S. Bagchi, and E. J. Coyle, in Vehicular Technology Conference (VTC), 2007.
  - **"Dynamic Application Management to address SLAs in a Virtualized Server Environment,"** G. Khanna, K. Beaty, A. Kochut, and G. Kar, in Network Operations and Management (NOMS), 2006.
  - **"Synchronization Attacks Against 802.11,"** G. Khanna, A. Masood, and C. N. Rotaru, in Network and Distributed System Security Symposium (NDSS) Workshop, Feb 2-4, San Diego, 2005.
  - **"Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Networks,"** G. Khanna, S. Bagchi, and Y. S. Wu, In Dependable Systems and Networks (DSN), pp. 795-804, Florence, Italy, 2004.
  - **"Data Dissemination Protocol to account for Node and Link Failures in Sensor Networks,"** G. Khanna, S. Bagchi, and Y. S. Wu, Fast Abstract Dependable Systems and Networks (DSN), 2003.

Thank You !!!

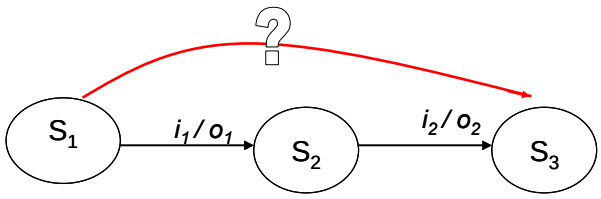
Backup Slides

Motivation for State Space Reduction

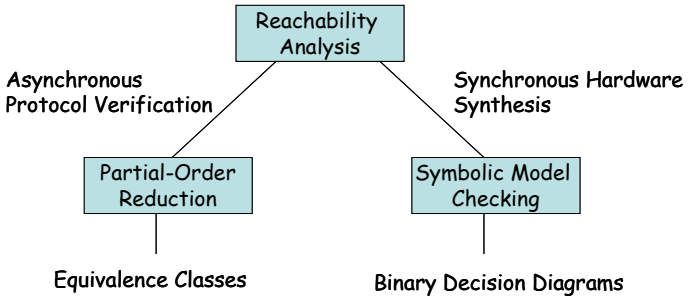
- Complex Distributed Applications
  - Large scale
  - Complex protocol
- Result: Large state space
  - Cause state space explosion for verification
- For a monitoring system
  - Not all transitions are valuable to a monitoring system
  - Internal transition of a protocol entities
  - Transition of protocol entities behind firewall
  - No rule associate with a particular transition

STD Reduction: Monitor Performance

- Monitor(s) has a rule base for verification
  - All states are not *verified*
- Some states might be internal because of the black-box model of the protocol entities
  - Internal States can cause Monitor to loose track
- STD must be reduced to circumvent these cases



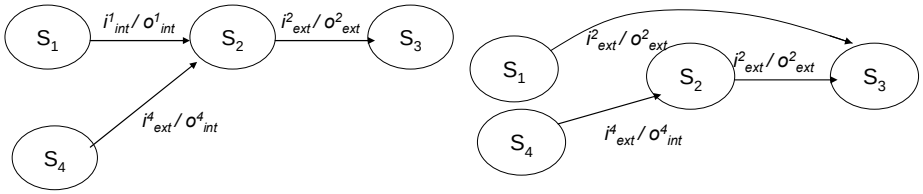
Related Work: State Space Explosion



- Simple example: consider a trace $s_1 \rightarrow_{\alpha 1} s_2 \rightarrow_{\alpha 2} s_3 \rightarrow_{\alpha 3} s_4$. If transition α_2 is hidden (i.e., internal) then partial order method will mark states s_3 and s_4 as unreachable and reduce both
- But a monitoring system may want to verify state s_4

STD Reduction : 2 Phase Reduction

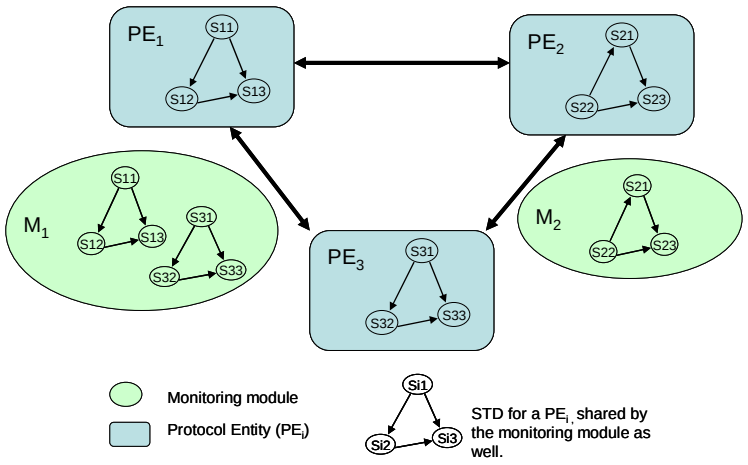
- Removing internal transitions
  - **C1**:*"If a state does not have any external transition edge, then remove that state and re-assign all the external outgoing transition edges of the reduced state".*
  - **C2:***"If a state exists which has both internal and external transition edges, we need to remove the internal messages and re-assign the external incoming and outgoing transition edges of that state".*



STD Reduction : Two Phase Reduction

- Removing rule less states
  - Reduces the amount of storage and computation needed at the detection process
  - Reducing the number of nodes that need to be traversed during the diagnosis process
- Analytically prove that reduction process does not affect the missed alarms or false alarms generated during the detection procedure
  - No modification to the rule structure or detection procedure
- Analyze the performance gains through actual test-bed experiments running the Monitor framework
  - Significant reduction in latency of detection and diagnosis

Solution Approach



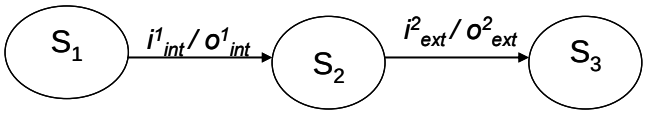
Removing Internal Transition

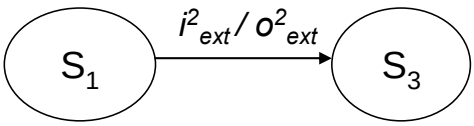
Removing internal transitions (STD→STD′)

- The transition is internal to the PE -> no externally visible message.
- The monitoring system is placed in a network location where the observation of the PE is not perfect
- There are firewall rules that block the monitoring system from observing this kind of transition.

Case 1

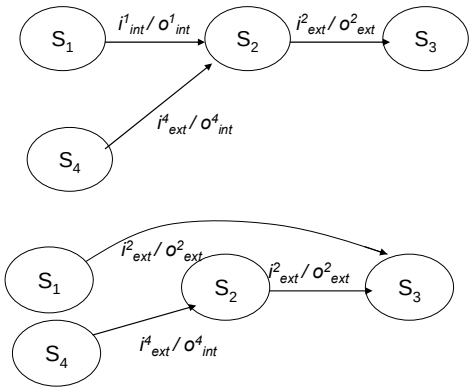
- **C1**: *"If a state does not have any external transition edge, then remove that state and re-assign all the external outgoing transition edges of the reduced state".*



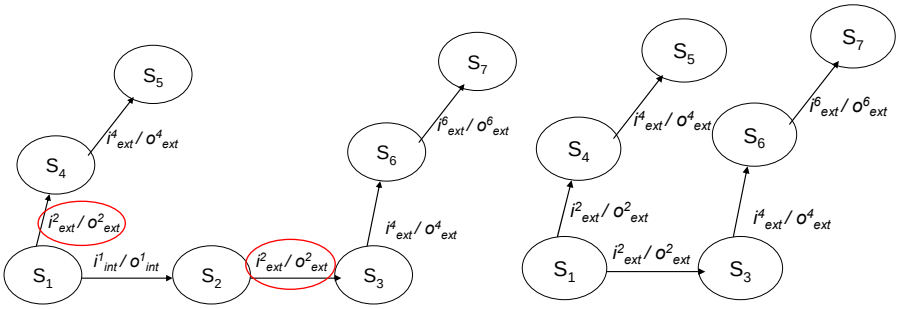


Case 2

- **C2:**"*If a state exists which has both internal and external transition edges, we need to remove the internal messages and re-assign the external incoming and outgoing transition edges of that state*".



Case 3: Non-deterministic transitions



Removing transition without rules

STD reduction due to ruleless states (STD′→STD″)

- Reduces the amount of storage and computation needed at the detetion process
- Reducing the number of nodes that need to be traversed during the diagnosis process

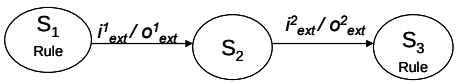
5 types of rules

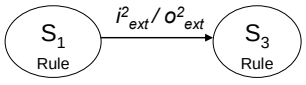
- **Type II:** *St* is the state of an object at time *t* : St ≠ St+Δ, if event Ei takes place at t
- **Type V:** This rule prevents a state transition from *Si* back to the same state within time β of first arriving at *Si*.

$$s = S_i \ \forall t \in (t_0, t_0 + \alpha) \Rightarrow s \neq S_i \ \forall t \in (t_0 + \alpha, t_0 + \beta);\ s.t. \beta > \alpha$$

Case 1

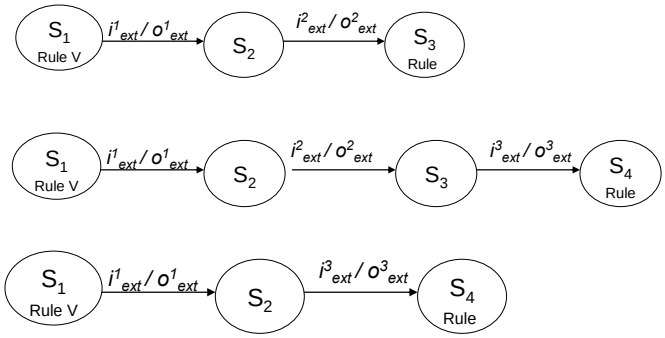
- **C1**: A state *Si* has no rule ∧ (∀*Sj* ∈ *predecessor*(*Si*)) *Sj* has no rule of type V ∧ *Sj* has no rule on transition edge *i1ext*/*o1ext*⇒ Remove *Si* and incoming transitions of *Si*.





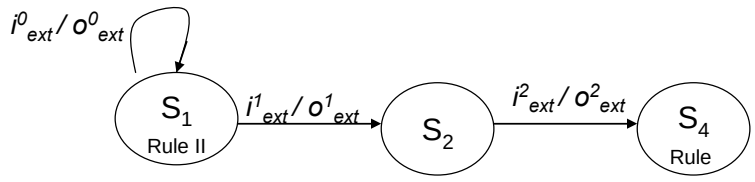
Case 2: Type V Rule

- **C2**: A state S_i has no rule and ($\exists S_j \in$ predecessor(S_i)) *s.t.* S_j has a rule of type V $\Rightarrow$ Do not remove S_i.



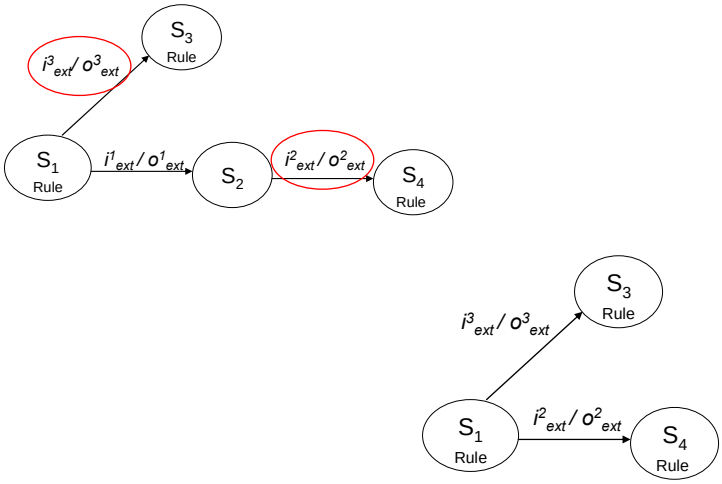
Case 3: Type II Rule

- A state S_i has no rule and ($\exists S_j \in$ predecessor(S_i)) *s.t.* S_j has a rule of type II $\wedge$ T_j is self-loop $\Rightarrow$ Do not remove S_i.



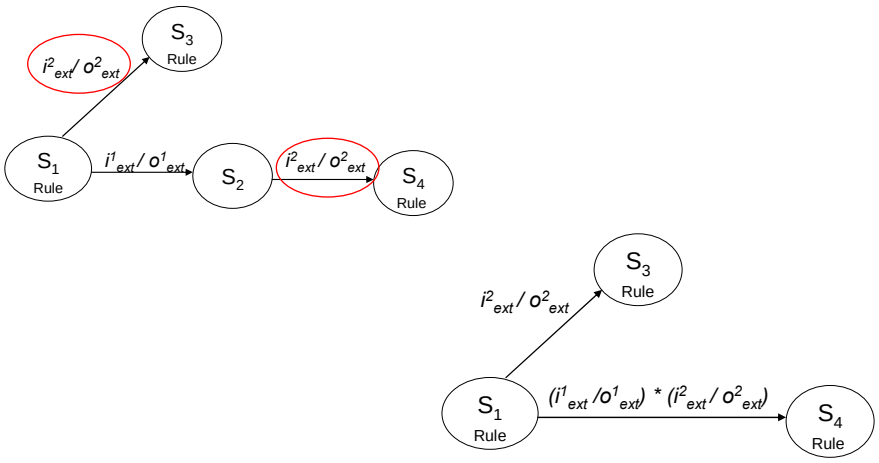
Example: Simple

- Example



PURDUE UNIVERSITY

Example: Complex edge



PURDUE UNIVERSITY

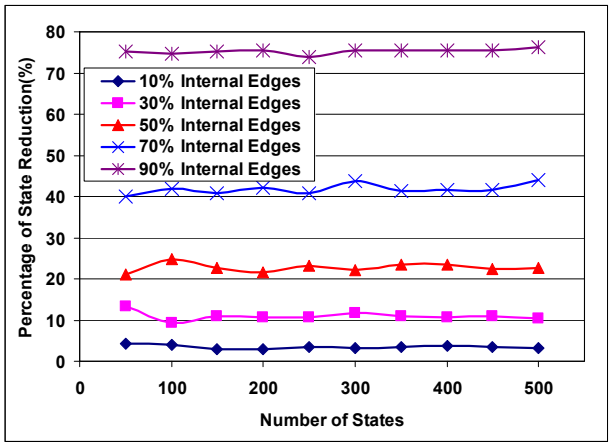
Experiments

- Developed an emulator system
  - generates a random STD
  - emulates PEs which performs transitions
- Divided the experiments into two parts:
  1) Static
     - Test the efficacy of the reduction mechanism by inputting several STDs which have internal states and rule less states.
  2) Dynamic
     - Test the latency of the Montior System

Emulator

- Performs two tasks:
  1. Generate a random state transition diagram via generating a random graph
     - Generates a random connected graph
     - Marks some of the transition edges E to be internal
     - Marks states to contain rules
  2. Emulate PEs which perform transitions according to the generated STD.
     - Emulates some PEs which exchange messages amongst each other
     - Via performing a *random walk* over the generated STD

STD' Reduction



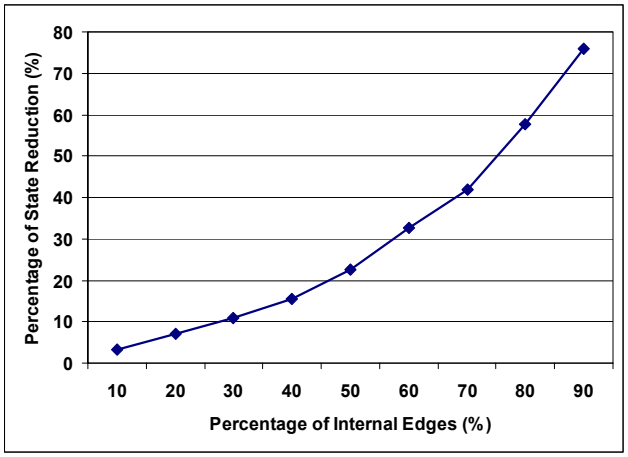
The STD contains $4V$ edges

percentage of state space reduction =(#states in STD – #states in STD')/ #states in STD

- We can see that for a fixed fraction of internal transition edges, the percentage of state space reduction remains constant with increasing state size.

PURDUE UNIVERSITY

STD' Reduction



- An increase in percentage of state space reduction with increasing percentage of internal transition edges.
- A state is removed only if all incoming transition edges are internal.

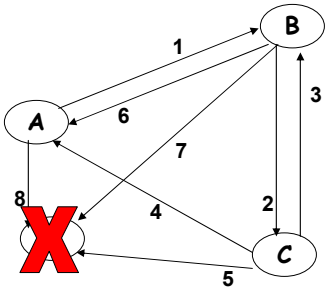
PURDUE UNIVERSITY

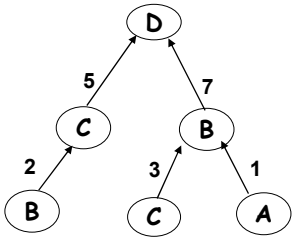
Diagnosis in a gist

- Monitor maintains a *causal graph* with events ordered according to the logical time
- c by building a Diagnosis 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 all nodes are not faulty then suspicion set is expanded to include the nodes which sent messages to nodes in A.

Causal Graph & Diagnosis Tree

4 PEs namely A, B, C and D exchange messages 1-8 amongst each other. The message number indicates the causal order i.e. message 1 precedes all of the rest of messages.





Diagnosis Tree

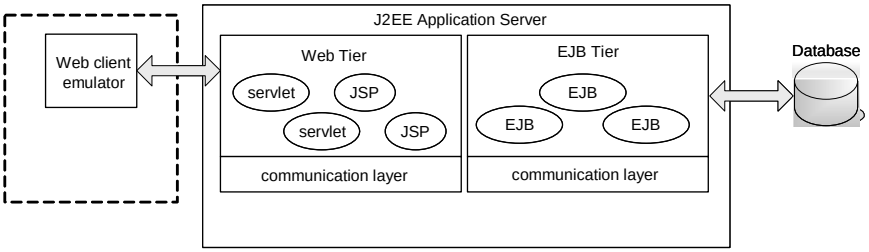
Message ID	Sender.LogicalClock , Receiver.LogicalClock
1	A.LC1, B.LC1

Failure Detection

- Monitor uses rule base to verify the observed messages
  - Generic architecture widely applicable through using specific rule base
  - Black-box, non-intrusive semantics
- Scalability is achieved through hierarchical design
- Automatic load balancing helps in making the system auto-configurable

Pinpoint's Approach to Diagnosis(1)

- Pinpoint is an approach developed by Chen et.al in 2002 to address problem determination in E-commerce system
  - Approach is generic can be applied in larger context
- It uses a dependency matrix describing the dependence of client transaction on web components
  - Components consists of EJBs, and servlets
- Internal and external failure detectors are used to determine success of client transactions



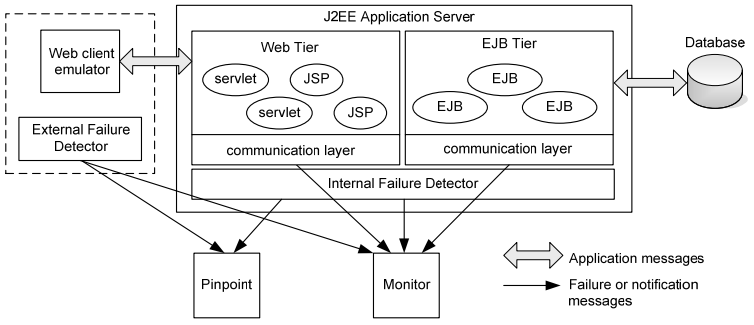
Pinpoint's Approach to Diagnosis(2)

- A failure column is added to the dependency matrix which contains the outcome of the transaction (failed or success)
- Pinpoint correlates the failures of transactions to the components that are most likely to be the cause of the failure
- Example:

Client Request ID	Failure	Component A	Component B	Component C
1	0	1	0	0
2	1	1	1	0
3	1	0	1	0
4	0	0	0	1

Comparison with Pinpoint: Experimental Set-up

- We implement the Pinpoint algorithm and create a e-commerce system as described in the Pinpoint paper
  - PetStore application deployed on JBoss application server
- Provide external and internal failure detectors to both Monitor and Pinpoint for a fair comparison of only diagnosis approaches

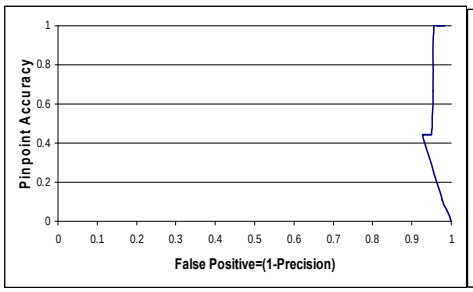


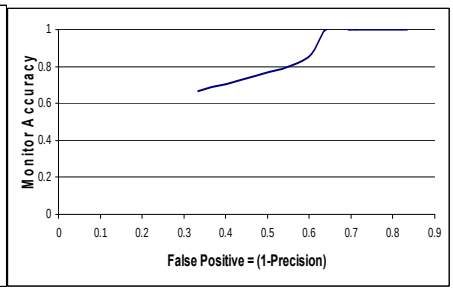
Experimental Set-up

- We use a total of 55 client transactions out of which 45 are unique.
- We chose 9 components, 6 EJBs and 3 servlets as our target components for fault injection
  - Examples: *AddressEJB, AsyncSenderEJB* etc.
- We perform 4 different type of fault injection into the components similar to Pinpoint
  - Declared Exception, Undeclared Exception, Endless call and Null call
- Similar to Pinpoint we use 1-component, 2-component and 3-component triggers for fault injection
  - In a 2-component trigger, a sequence of 2-components is determined and whenever the sequence is touched during a transaction, the last component in the transaction is injected with the fault
- Accuracy and Precision metrics are compared
  - Predicted Fault set is {A, B, C}, but only {A} has a fault then accuracy is 100% but precision is 33.3%.

1-Component Faults

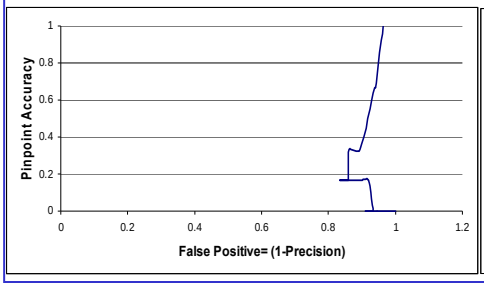
- All 4 types of fault injections are performed on each of the 9 components
- Pinpoint has high false positives rates but the accuracy eventually reaches 1. In contrast, the Monitor has a much higher accuracy keeping a low false positive rate. Monitor's accuracy also reaches 1 but at a much lower value of false positives (0.6) as compared to Pinpoint (> 0.9)

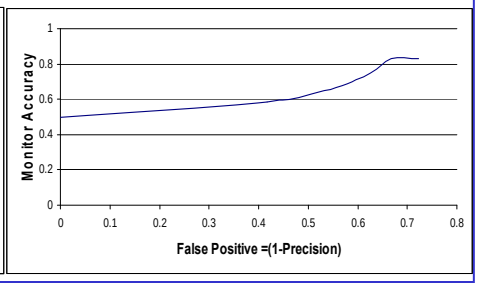




2-Component Faults

- Monitor outperforms Pinpoint in the 2-component fault injection.
- One can see that accuracy reaches a maximum of 0.83 compared to 1.00 in 1-component injection
- In 2-component fault injection the accuracy is not as high as 1-component faults because the number of diagnosis instances are far less

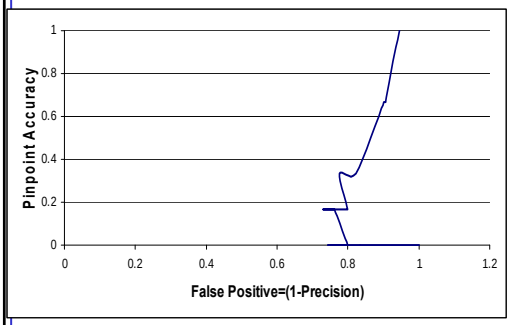


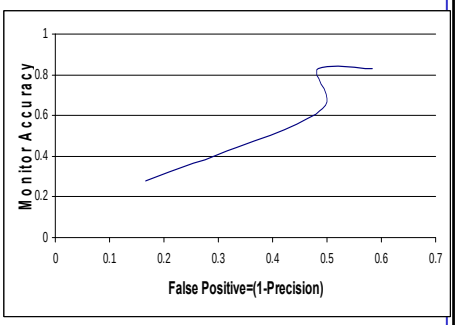


PURDUE UNIVERSITY

3-Component Faults

- The relative behavior of the Monitor's diagnosis algorithm and Pinpoint's approach remains the same in 3-component faults as well
- Monitor achieves low false positive rates which providing a much higher accuracy than Pinpoint

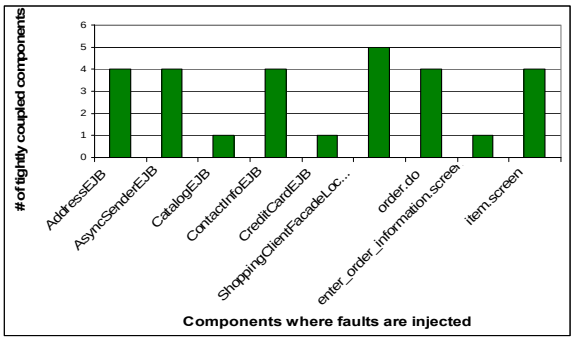




PURDUE UNIVERSITY

Behavior of Components

- For the success of Pinpoint's algorithm, the components should behave independently
- Presence of tightly coupled components can cause the clustering approach to suffer
- We observe some coupling in the target components as well
  - PetStore software is written in such a manner that required coupling of multiple components



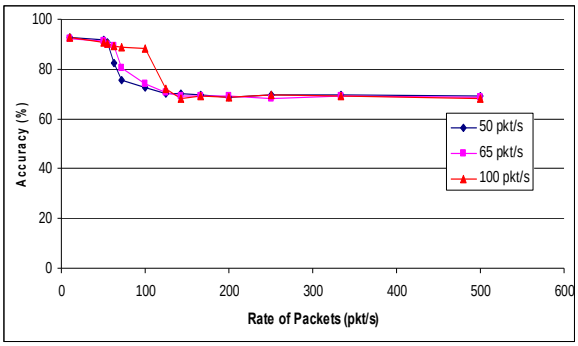
PURDUE UNIVERSITY

Comparison with Pinpoint: Conclusions

- Monitor's probabilistic diagnosis is compared with Pinpoint approach to diagnosis
  - Pinpoint is implemented
  - Monitor and Pinpoint are used to diagnose failures in an e-commerce set up
  - Experimental set-up is made close to the Pinpoint paper and same fault injection are performed
- Monitor outperforms Pinpoint by achieving higher accuracy for the same precision values
- Monitor achieves lower precision numbers for 1, 2, and 3-component faults thus providing a low false positive rates

PURDUE UNIVERSITY

Effects of Varying R_{th} : Accuracy



- Beyond R_{th} Monitor switches from Monitor-HT to Monitor-S
- Difference in characteristics of the curve around R_{th} provides the system administrator a useful tuning parameter

Example of Detection Rules on TRAM

- *T R4 S4 E11 30 500 5000 S4 E2 1 8 4000 7000*: The rule has a precondition to check data packets (*E11*) arrival within 5000msec. This causes the post condition that at least one ack(*E2*) (between 1 and 8) must be sent
- *T R3 S5 E13 0 5 5000*: This rule ensures that the number of re-affiliation packets (E13) is no more than 5 within 5000ms in state S7
- *T R3 S0 E1 10 30 5000*: This rule of type 3 checks for the hello packet(*E1*) rate. The E1 message count should be between 10 and 30 for the next 5000 msec
- *T R4 S0 E10 1 4 1000 S3 E8 1 2 3000 4000*: Head Adv.(*E10*) messages should be eventually followed by Accept message(*E8*)

Rule Base

- Monitor is provided with Normal Rule Base for detection and Strict Rule Base for diagnosis

- A few examples of SRB rules
  - *S1 E11 1 S3 E11 30 1* : A single data packet must be followed by 30 more data packets
  - *S6 E1 1 S6 E9 1 1* : Hello Message must be followed by a Hello Reply
  - *S1 E11 1 S2 E11 1 1* : A repair head must send out each data packet which is received
  - *S0 E15 1 S1 E14 1 1*: A receiver sends a Head-Bind message then it should receive multiple Head-Ack packets

Related Research

Observer Systems

- Near identical approach is presented through the *observer* system
  - Monolithic Entity, Formal verification required *[Diaz '94]*
- Approach using Communicating Finite State Machines (CFSM) *[Seviora DSN '02]*
  - Global correctness is assumed via individual local interaction verifications
  - Claim to eliminate the state space explosion problem

Diagnosis

- White-Box diagnosis
  - Have access to the internals of the system [Gruschke 1994], [Sanders 2005]
  - Use active probing to infer the problem
  - Use of heavy *instrumentation*
  - Embedding event generators in the application

Related Research

- Multiprocessor diagnosis
  - Deterministic diagnosis approach: First diagnosis approach by PMC *[Preparata et.al. 1967]*
  - Several testing graph based approaches have been proposed with variation in number of tests and graph structure
  - *t*-diagnosable *[Hakimi 74]*
  - Hierarchical testing algorithm was proposed requiring $(\log N)^2$ testing rounds *[Nanya 98]*
  - Probabilistic diagnosis by Fussel and Rangarajan [1989], followed by *[Kang and Lee 1994]*
  - Assume no distinction between the protocol entities and testing entities
  - Employ explicit tests to the entities
- Debugging in Distributed Applications
  - Industrial research has focused on problem determination on distributed applications like eCommerce. For e.g. dependency graph *[Kar, Hellerstein]*
  - Different approaches incorporating probing, dependency analysis and adaptive diagnosis have been proposed
  - ***Black-Box***: It aims to find a find causal relationship between the RPC messages
    - Stops at finding the causal relationship
    - Offline Method which involves heuristics *[Aguilera 2003]*

Temporal rules

Type I:

$$S_p = \text{true for } T \in (t_N, t_N + k) \Rightarrow S_q = \text{true for } T \in (t_I, t_I + b)$$

Type II:

S_t is the state of an object at time t : $S_t \neq S_t + \Delta$, if event E_i takes place at t

Type III:

$$L \leq |V_t| \leq U \ (t_i, t_i + k)$$

Type IV:

$$\forall t \in (t_i, t_i + k)\ L \leq |V_t| \leq U \Rightarrow L' \leq |B_q| \leq U',\ \forall q \in (t_n, t_n + b)$$