Stateful Detection in High Throughput Distributed Systems

Gunjan Khanna, Ignacio Laguna, Fahad A. Arshad, Saurabh Bagchi

Dependable Computing Systems Lab
School of Electrical and Computer Engineering
Purdue University

PURDUE UNIVERSITY

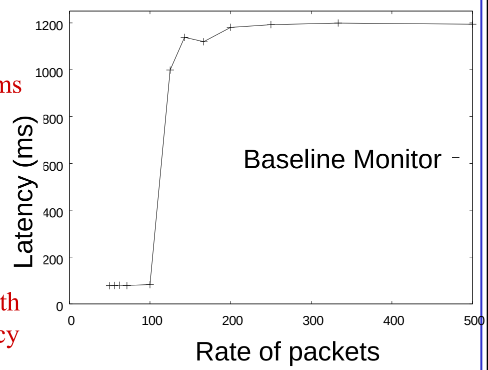
Outline

- Motivation
- Monitor Based Error Detection
- Sampling Approach
- Experiments and Results.
- Related Work
- Conclusions and take-away lessons

PURDUE UNIVERSITY

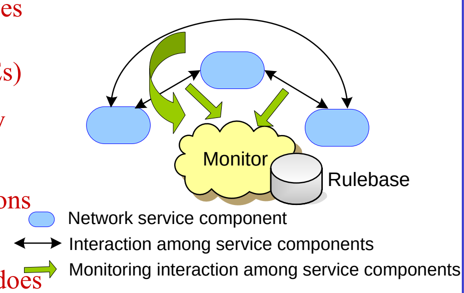
Motivation

- Ever increasing bandwidth has led to proliferation of distributed applications with high throughput streams
- Error detection framework must therefore handle high throughput streams
- Our previous detection mechanism breaks beyond a particular incoming packet rate
- The goal is to push the knee to right with graceful degradation of detection latency and accuracy



Monitor Approach to Detection

- The approach follows an observer-observed model
- The observer (the Monitor) only observes external message interactions between components called protocol entities (PEs)
- The Monitor maintains a set of anomaly based rules to verify the PEs
- The Monitor estimates the state transitions of the PEs
- On an incoming message, the Monitor does the appropriate state transition and matches the state specific rules in the rule base
- A violation of the rule would flag an error leading to a detection of failure



Design Goals

- Online mechanism enforcing low latency and high accuracy
  - Graceful degradation of latency and accuracy
- Treat protocol entities as Black-box
  - Non-intrusive approach
  - Operate asynchronously with respect to application
- Monitor should be executable on off-the-shelf hardware
  - Should not have large memory footprint
  - The computation should scale slowly with the number of PEs
- Stateful approach should be followed
  - Natural errors in systems are stateful
  - Example: Failures in Windows NT
  - Example: Failure prediction in cycle-sharing systems

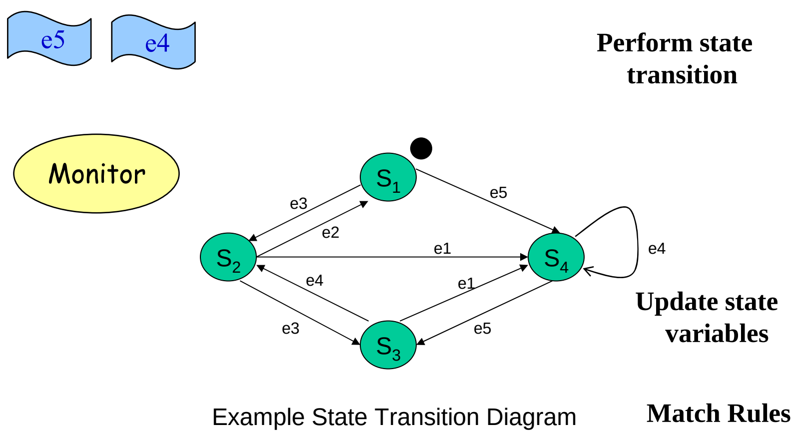
Detection Framework

Step1: On observing a message, perform state transition and if required, update variables corresponding to PE`s STD

Step2: Perform rule matching for the rules associated with the particular state and message combination

Step3: Monitor flags an error if rules don`t match

Detection Framework



Example State Transition Diagram (STD)

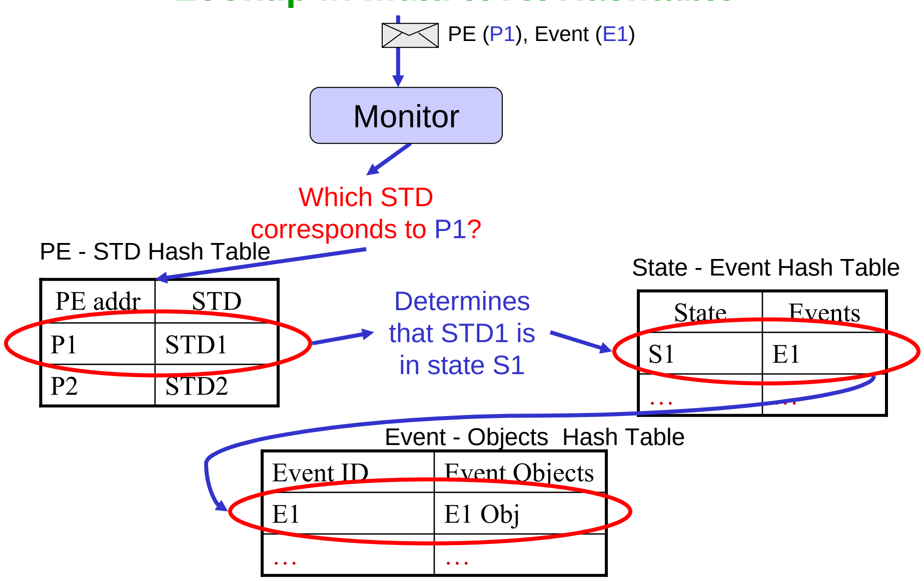
Rule Matching

- **Monitor-Baseline** has linear structures that it needs to traverse for rule matching
- Rules are 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 data 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*)

Solution: Improve Per Message Processing (Monitor-HT)

- Rationale: Make processing of each incoming message more efficient
- Solution Approach:
  - Provide efficient look-up using hashtables
  - Eliminate duplicate copies of the state variables
- State Transition Diagram is organized in a multi-level hashtable
  - Monitor-H has a constant order look-up while Monitor-Baseline was linear in the number of PEs being verified and number of events in each state

Lookup in multi-level Hashtable



Efficient Rule Matching – *Monitor-HT*

Data

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

Previous Approach

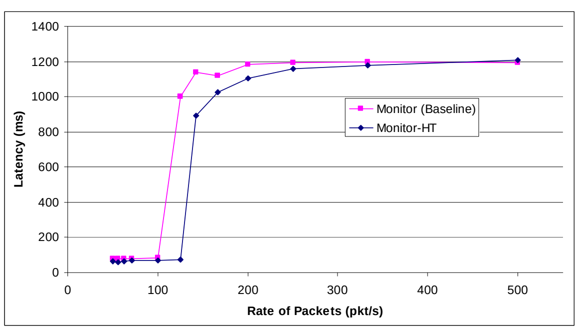
Data

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* $\times$ *Event ID* tuple is only incremented once
  - Single copy update of state variables.

Monitor-HT versus Monitor-Baseline



- We compare the latency of detection of Monitor-Baseline and Monitor-HT. 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

Solution: Sample Messages for Detection

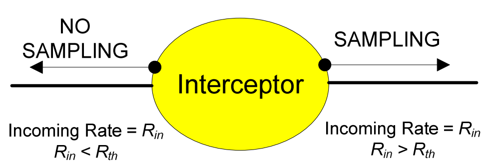
- Rationale:
  - Monitor-HT still has to perform a minimum constant amount of work for every incoming message.
  - It gets overwhelmed when the message rate is too high
- Solution Approach
  - Modify Monitor-HT to reduce the incoming workload
  - Sample incoming messages to perform matching on a subset
- Instead of processing every message, sample the incoming messages **(Monitor-S)**
  1. When do we do sampling ?
  2. How do we sample?
  3. How do we handle state non-determinism?

Slide 13/26

PURDUE UNIVERSITY

When do we Sample?

- Assume Monitor-Baseline achieves a desired latency and accuracy uptil a rate of incoming messages R_{bp}.
  - Choose a threshold R_{th} such that $R_{th} < R_{bp}$
- When the incoming rate R_{in} is such that $R_{in} > R_{th}$, Monitor-S switches to sampling mode of operation
- Design tradeoff:
  - R_{th} is far below R_{bp}: Inefficient use of Monior resources
  - R_{th} is very close to R_{bp}: Small spike can make the system unstable



Slide 14/26

PURDUE UNIVERSITY

How do we Sample?

- We choose uniform random sampling: rate of sampling is dependent on the rate of incoming messages
  - 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
- 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
- Scale the constants in the rules by a factor of R_{th} / R_{in}
  - Original rule — "Receive 10 Acks in 100 sec"
  - Rule modified due to sampling — "Receive 10.(R_{th} / R_{in}) Acks in 100 sec"

PURDUE UNIVERSITY

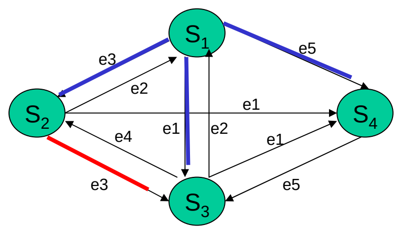
How do we handle Non-Determinism ?

- Dropping a message can cause Monitor-S to lose track of the current state of the PE
- Instead of keeping a single current state for each PE being verified, keep a vector of possible states the PE may be in
  - $\hat{S} = \{S_1, S_2 \ldots S_K\}$
- 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 is expensive. So Monitor-S pre-computes state vectors offline

PURDUE UNIVERSITY

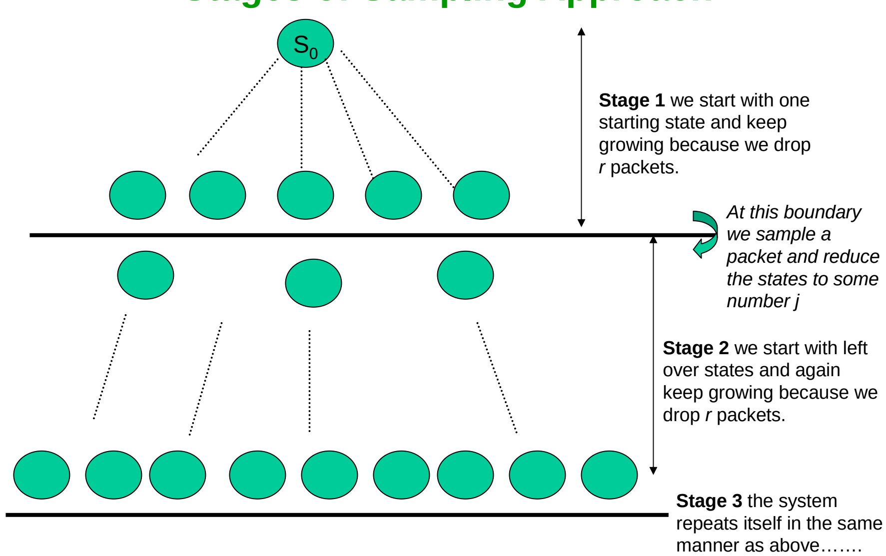
How do we handle Non-Determinism ?

- Size of the state vector does not keep growing
  - Bounded by the total number of states
  - Sampling of a message causes state vector size to reduce since message is only possible in a few of the states of the state vector
- 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): $\hat{S} = \{S_2\}$



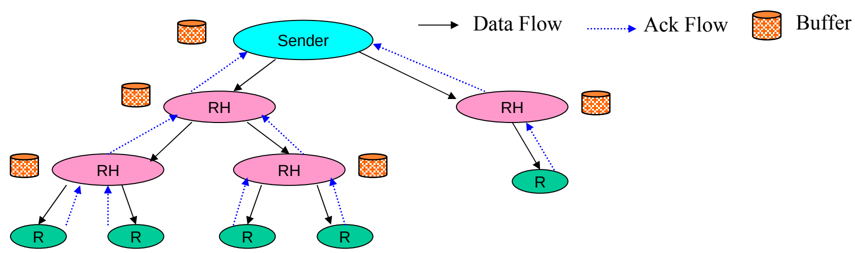
Example State Transition Diagram (STD)

Stages of Sampling Approach



Demonstration: TRAM

- We demonstrate the use of the Monitor on TRAM, *a tree-based reliable multicast protocol.*
- TRAM consists of a single sender, multiple repair heads (RH), and receivers. It ensures reliability of message transfer in case of node or link failures and message errors.
- We emulate TRAM protocol, where sender and receiver are the PEs being verified by the Monitor in all experiments.

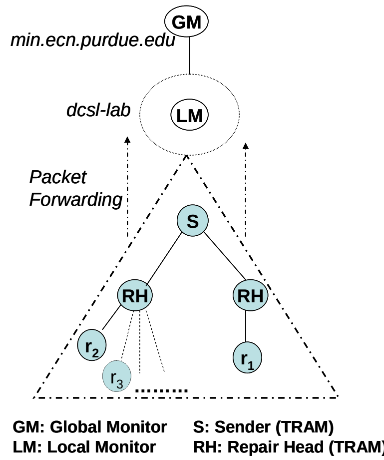


Failure Injection

- We perform *random fault injection* in the header of the emulated TRAM messages to induce failures
- We randomly choose a header field and change it to a randomly selected value, emulating protocol errors
  - Say, too many NACK messages are sent by the receiver
- To mimic faults close to reality, a burst length is chosen since TRAM is robust to isolated faults
  - A PE to inject is chosen (sender, RH or receiver) and faults are injected for a burst length of 500ms after every 5 minutes.

Topology and Metrics for Experiments

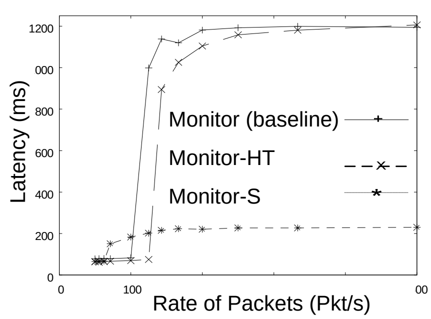
- *Accuracy* = (1- % of missed detections)
- Fault injections undetected by the Monitor are called *missed detections*
- *False detections* are errors flagged by Monitor but which do not affect TRAM entities
- *Latency* is measured as the time from the instantiation of a rule to the time when the rule matching is completed

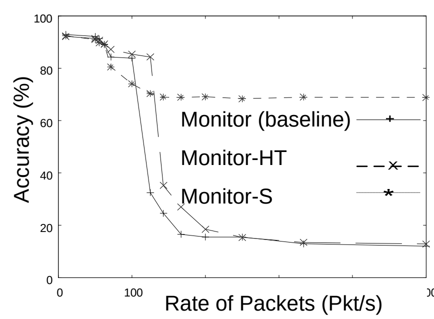


Physical Topology of the TRAM emulator and the Monitor

PURDUE UNIVERSITY

Results: Latency and Accuracy with packet rate (R_{in})

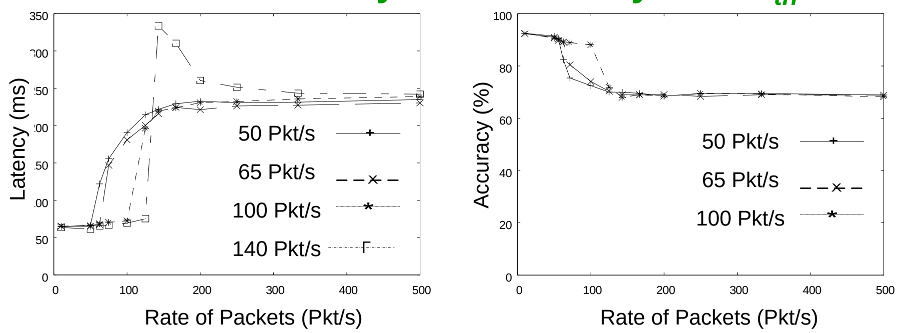




- Montior-Baseline and Monitor-HT break causing a knee. Monitor-HT`s knee is beyond Monitor-Basline`s knee
- Monitor-S shows a relatively smooth degradation in both accuracy and latency results
  - It is able to adjust the workload based on the rate
- Montior-S gives a low latency at the cost of small reduction in accuracy at high input rates
  - Accuracy suffers a little because of non determinism

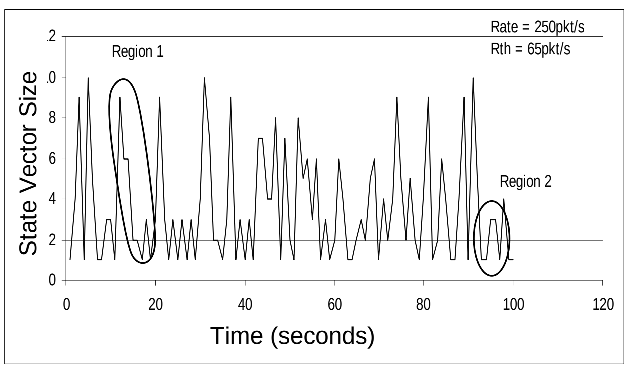
PURDUE UNIVERSITY

Results: Latency and Accuracy with R_{th}



- For $R_{th} < R_{bp}$ (R_{bp}=125 pkt/s), latency increases slowly before smoothing to a constant value
- For the curve of R_{th} = 140 pkt/s, the jump in latency is because the incoming rate is greater than R_{bp} and Monitor switches from sampling to non-sampling modes after the breakpoint
- The jump in latency translates to a sharp drop in accuracy in the accuracy plot
- A low R_{th} (50 pkt/s, 60 pkt/s) kicks in sampling early reducing accuracy. For R_{th} =100pkts/s the accuracy is high for a larger incoming rate

Results: Variation of State Vector Size



- In Region 1, $|\hat{S}|$ drops in steps from 9 to 6 and finally to 1. The drop in $|\hat{S}|$ is because of the unique possibility of the sampled event in only some of the states
- In Region 2, $|\hat{S}|$ increases from 1 to 3 because of a message drop

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, DSN`04
  - Restricted to the domain and focused on accuracy
- Detection in Distributed Systems
  - Heartbeats, watchdogs DSN `00
  - Detection of Failures using event graphs

Conclusions

- We developed a stateful detection mechanism that can scale to a high data rate of the application protocol
- We extend an existing detection approach (Monitor-Baseline) by identifying in-efficiencies in the rule matching process to reduce per packet processing overhead at the Monitor
- We use a sampling approach to reduce the number of packets being processed
- Take-Aways
  - Detection mechanism should do minimal per-packet processing
  - Sampling should be done only when needed (R_{in} close to R_{th})
  - Low sampling rate could lead to state space explosion