

Self Checking Network Protocols: A Monitor Based Approach

Gunjan Khanna, Padma Varadharajan, Saurabh Bagchi

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



<http://shay.ecn.purdue.edu/~dcs1>

Outline

- Motivation
- Monitor Approach
- Monitor Architecture
- Hierarchical Monitor approach
- Experiments and Results
- Other Approaches
- Conclusions

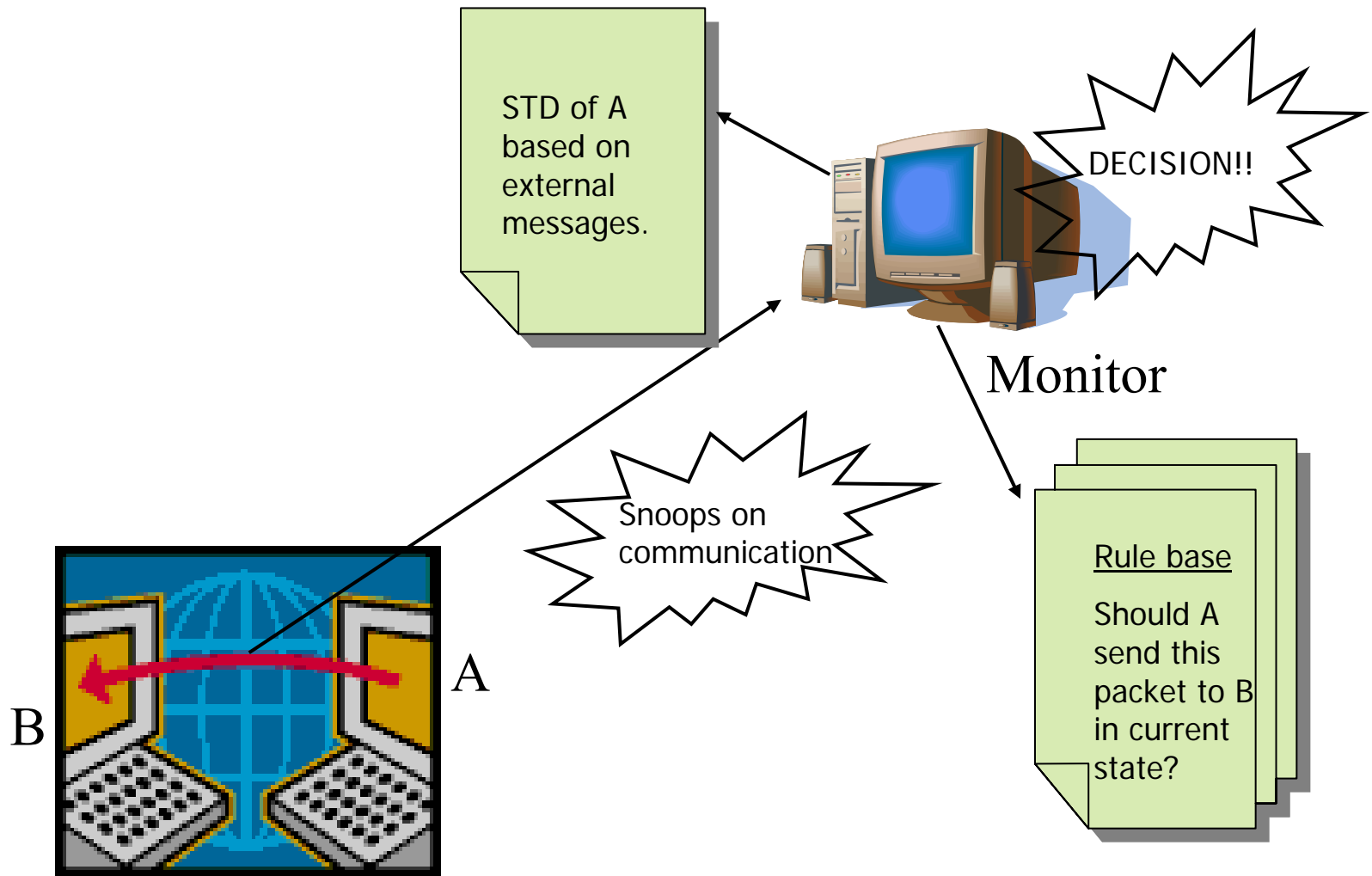
Motivation

- Wide deployment of high-speed networks has made distributed systems ubiquitous
- Infrastructure facing increasing threat of dependability outages
 - Natural failures
 - Malicious attacks
- Catastrophic consequences for downtime
 - Mean loss of revenue for distributed system downtime - \$1.01M/hour
 - In safety critical applications, loss of human lives
- We are focusing on the problem of detection of *disruptions*
 - Fast enough that faulty components can't communicate outside

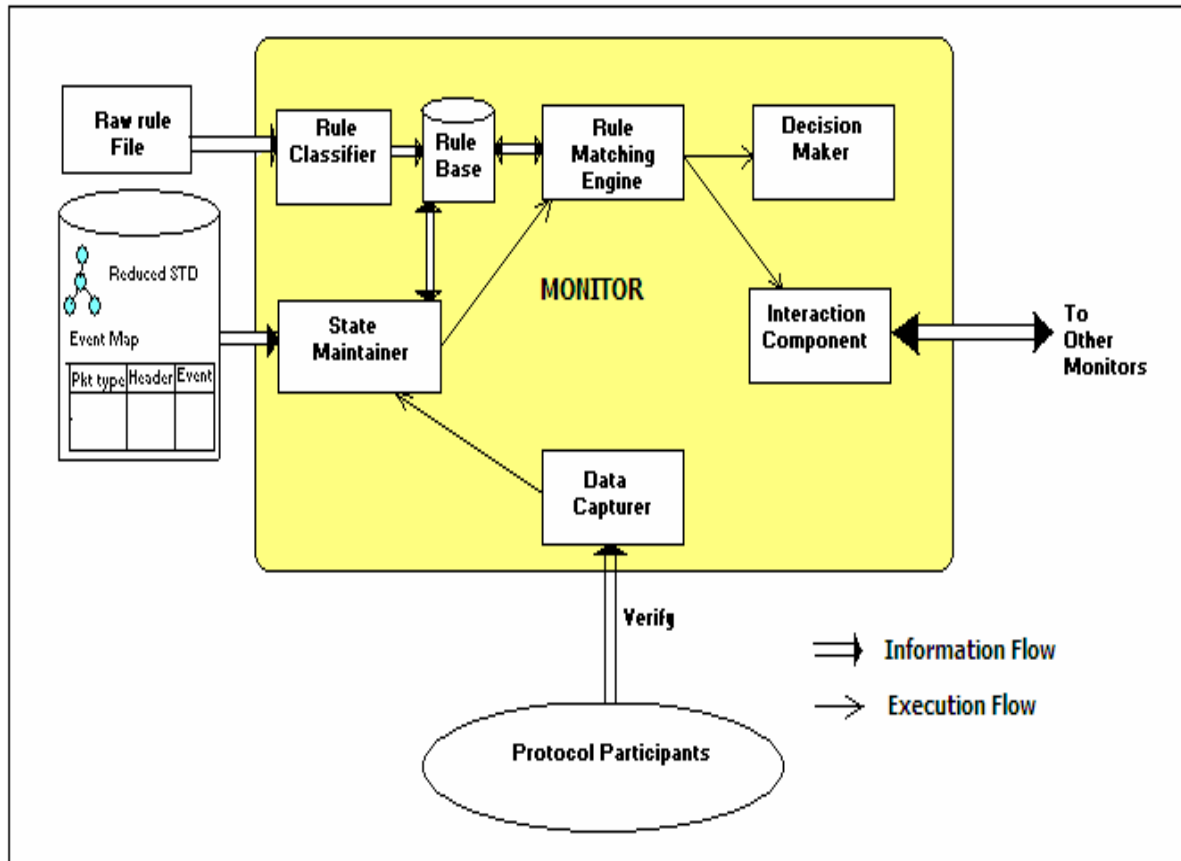
Challenges for Detection

- Detection infrastructure should be non-intrusive
- Applications are often blackbox
 - Legacy codes with non-availability of source code
- Large scale systems running into tens of thousands of nodes
- Systems often have soft real time guarantees
- Need for generic architecture

Monitor Approach



Monitor Architecture



Data Capturer:

Snoops over communication between PEs.

State Maintainer:

Contains event definitions & reduced STDs.

Flags rule matching based on State×Event

Rule Classifier:

Decides if rules are to be matched at current monitor.

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
- 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

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:

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

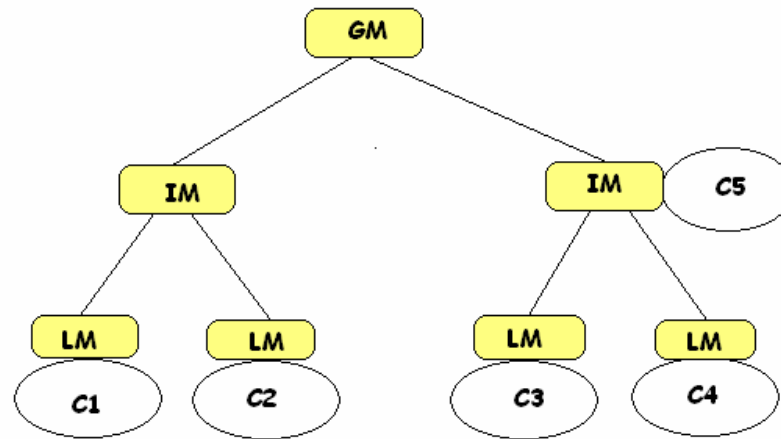
Type IV:

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

Rule Matching Engine

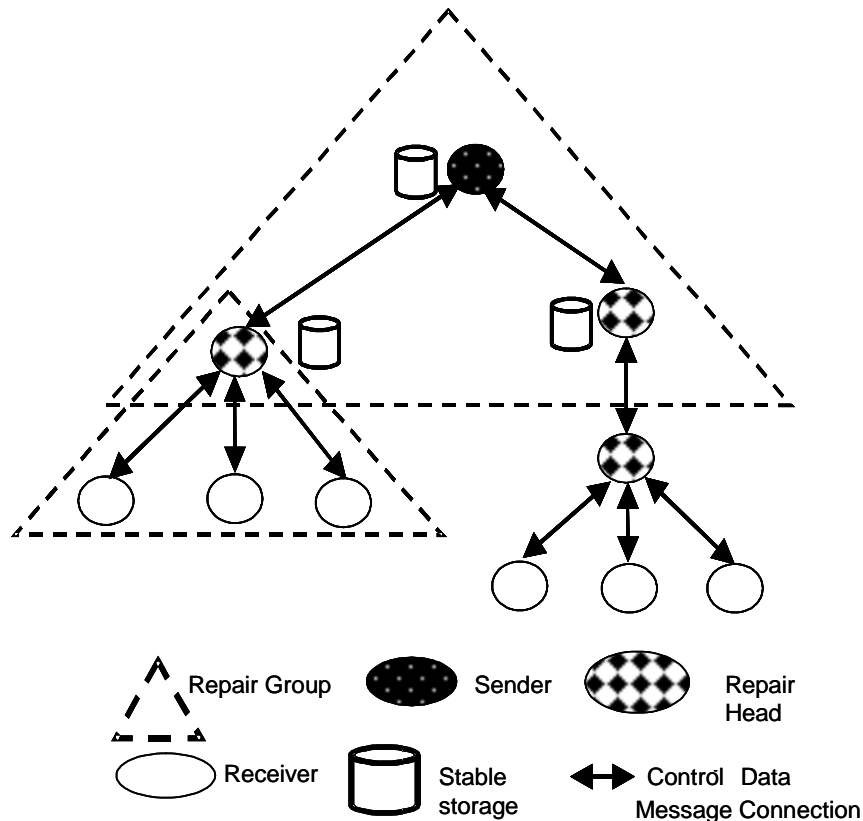
- Combinatorial rules translated into expression tree
- Rule matching done by traversing tree.
- Optimization - Previously computed value & list of operands in sub tree stored at each node.
- Two time scales for temporal rule matching – capture value of state variable, use value for rule matching.
- Optimizations for temporal rule matching
 - Fast hash table based lookup when events arrive
 - Thread pools for concurrency
 - Two separate thread pools for variable copying and matching
 - Categorization adds efficiency

Hierarchical Monitor Approach



- Removes single point of failure or performance bottleneck
- Adds accuracy and coverage to detection
- Increases redundancy
- Higher level Monitors see few messages from Local Monitors
- These messages may be aggregate messages (*e.g.*, count of the number of events) or direct messages from the PEs

Workload



- Monitor demonstrated on a streaming video application running on a reliable multicast protocol called TRAM.
- TRAM is hierarchical tree based
- Nodes in TRAM tree – sender, receiver, RH.

Examples of TRAM Rules

- **Combinatorial Rule:**

- The data rate at a receiver should be between M_{IN} and M_{AX} (specified as configuration parameters to the reliable multicast service)

- **Temporal Rule:**

- *TR3 S4 E12 0 5 5000*: The number of nacks in a period of 5000 ms should be less than 5
- *TR3 S1 E15 0 16 5000*: This is a global rule. The number of nacks seen globally in a period of 5000 ms should be less than 16. This rule is for the experimental configuration with 4 PEs under the GM.
- *TR2 S1 E11 50*: The state of the receiver should not remain the same 50 ms after receiving a data packet.

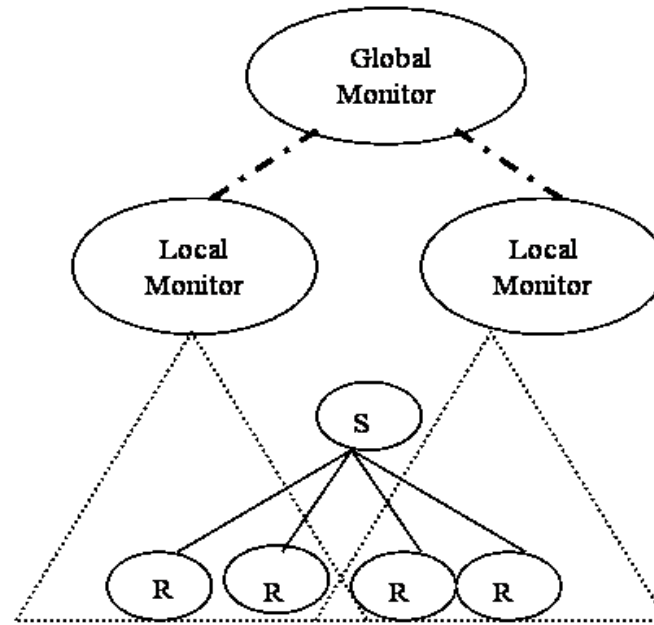
Error Injection, Experimental Setup

- MPEG-2 video stream with single server, multiple clients
- Minimum data rate – 20 KB/sec, Max data rate – 40 KB/sec
- Error injected into header of TRAM packet before sending, receiver actively forwards packet to Monitor
- 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)
 - Shorthand (NE; NC; DE)

Single Level Monitor Results

- Overall Monitor accuracy is 84.37%.
- Monitor accuracy very high for DE, but drops for (E; NC)
 - Very fast exception raising by protocol.
- In LR (**L**oose client, **R**andom injection), missed alarms mostly owing to Data→Ack packet conversion.
- In LD, increase in (E; C) errors, false alarms eliminated.
- In LS, more DE than in LD, low false alarms.
- Drop in coverage from loose client to tight client (87.2% to 81.6%)
 - Receiver checks data rate more frequently while Monitor latency remains same.

Hierarchical Monitor Experimental Results



- False alarm rate remains same
- Overall accuracy of 90.97%, 7% more than in the single Monitor case
- Significant improvement in LD case
- Global rule preemptively catches failure cases, owing to aggregated DE rule

Related Work

- **Formal specification of application behavior**
 - Extended State Machines [*Danthine, IEEE Trans. on Comm. '80*]
 - Temporal logic actions [*Lamport, TOPLAS '94*]
 - Petri Net based models [*Diaz, TOSE '91*]
- **Detection of crash failures**
 - Heartbeats, failure detectors etc.
 - In-built fault tolerant algorithms [*Schwartz, ToN '95; Hiltunen, SRDS '95*]
- **Detection using event graphs or CFSMs for restricted classes of faults** [*Wu ICPADS '97, Peng ICCCN '95*]
- **Two systems with similar goals and assumptions**
 - Observer – Worker system [*Diaz TOSE '94*]
 - Compositional approach, specifications using CFSMs [*Seviora DSN '02*]

Lessons Learnt

- Fast detection is possible by observing only external message exchanges
- Rule base creation is the labor intensive operation
- Structuring rule base into temporal rules (4 types) and combinatorial rules aids fast detection
- Hierarchical architecture helps scalability, latency, and coverage
- Tested on streaming video application using reliable multicast
 - Showing coverages of 84% and 91% for single and 2-level

Future Work -

- Dynamic environment where Monitors, PEs come and go
- Diagnosis in Monitor infrastructure.

That's all! Questions and comments?

