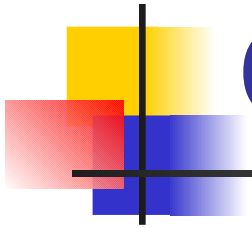
Dynamic Adaptation of Temporal Event Correlation Rules

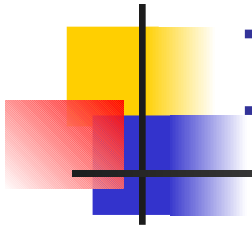
Rean Griffith‡, Gail Kaiser‡
Joseph Hellerstein*, Yixin Diao*

Presented by Rean Griffith
rg2023@cs.columbia.edu
‡ - Programming Systems Lab (PSL) Columbia University
* - IBM Thomas J. Watson Research Center



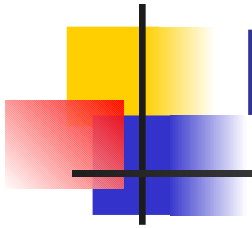
Overview

- Introduction
- Problem
- Solution
- System Architecture
- How it works – Feed-forward control
- Experiments
- Results I, II, III
- Conclusions & Future work



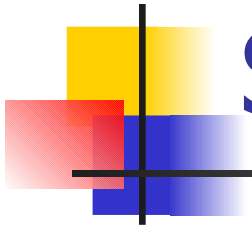
Introduction

- Temporal event correlation is essential to realizing self-managing distributed systems.
- For example, correlating multiple event streams from multiple event sources to detect:
  - System health/live-ness
  - Processing delays in single/multi-machine systems
  - Denial of service attacks
  - Anomalous application/machine-behavior



Problem

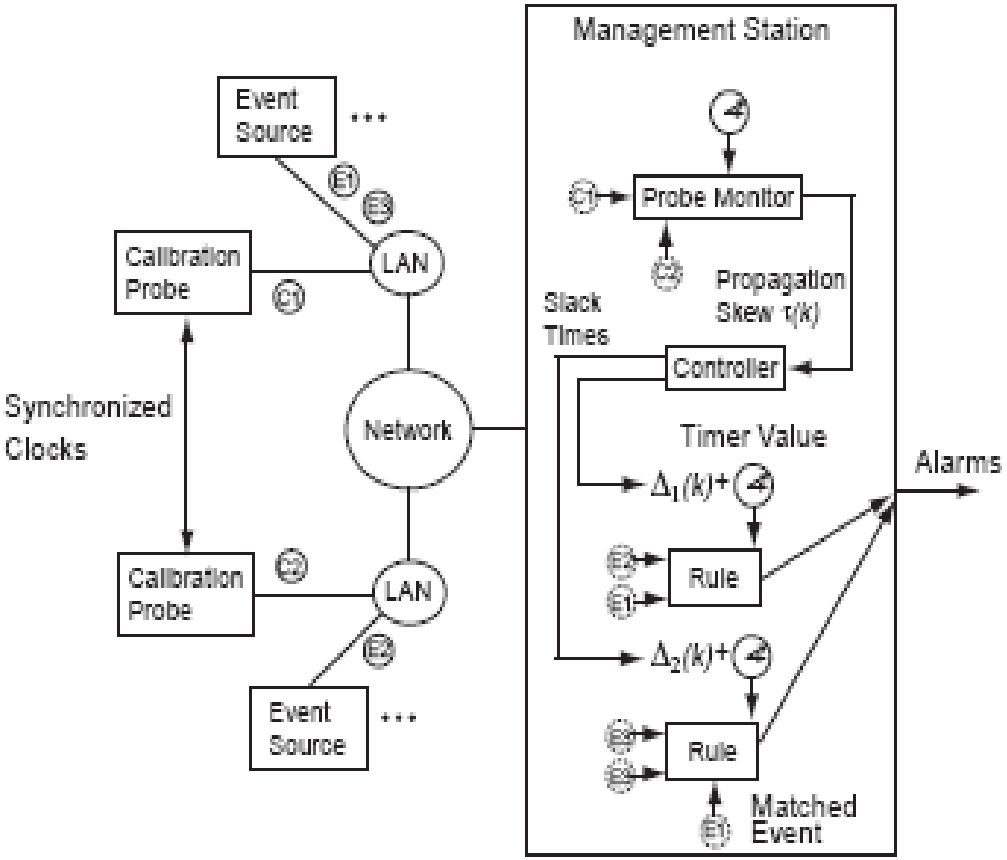
- Time-bounds that guide event stream analysis are usually fixed. Based on “guesstimates” that ignore dynamic changes in the operating environment
- Fixed time-bounds may result in false-alarms that distract administrators from responding to real problems.
- Issues with client-side timestamps (even with clock synchronization).



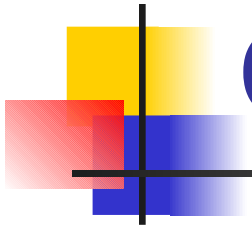
Solution

- Use time-bounds as the basis for temporal rules, but introduce an element of “fuzz” based on detected changes in the operating environment.
- To detect changes in the operating environment introduce Calibration Event Generators which generate sequences of events (Calibration frames) at a known resolution.
- Use the difference in the arrival times of calibration events to determine the “fuzz” to use.
- Only time-stamps at the receiver count.

System Architecture



How it works – Feed-forward Control



- Use the difference in the arrival times of calibration events within a calibration frame (less the generator resolution) as an observation of "propagation skew".
- Record last N observations of propagation skew.
- Sort these observations and use the median as the "fuzz" to add to timer rules
- Using the median prevents overreaction to transient spikes.

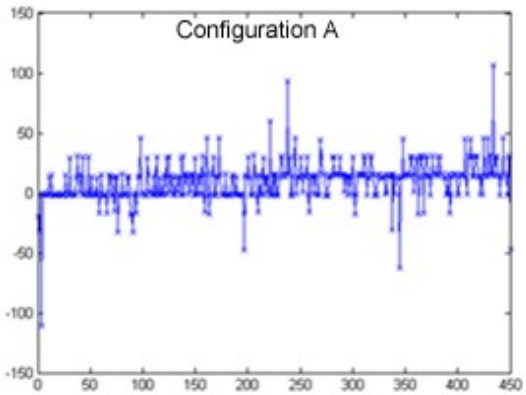
Experiments

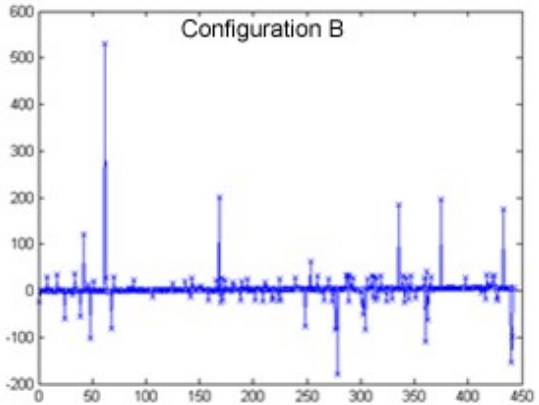
	Linux 2.6 3GHz, 1GB RAM	Linux 2.6 3GHz, 1GB RAM	Windows XP SP2, 3GHz, 1GB RAM	Linux 2.6 3GHz, 1GB RAM
Configuration A 3-machine	Calibration Event Generator	Siena Event Router	Event Distiller	N/A
Configuration B 3-machine	Calibration Event Generator	Siena Event Router	N/A	Event Distiller
Configuration C 2-machine	Calibration Event Generator	N/A	Siena Event Router + Event Distiller	N/A
Configuration D 2-machine	Calibration Event Generator	Siena Event Router + Event Distiller	N/A	N/A

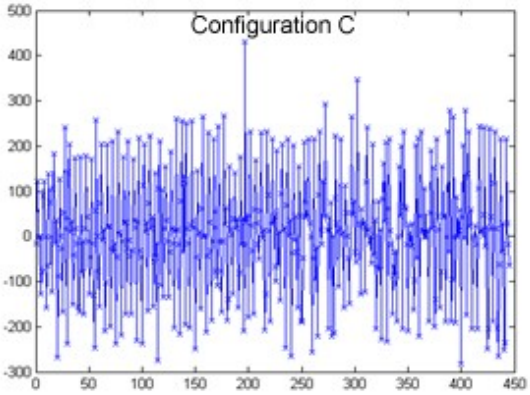
Results I – Propagation Skews

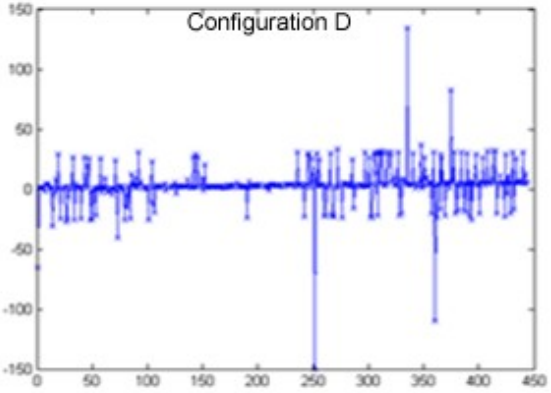
3-machine

2-machine







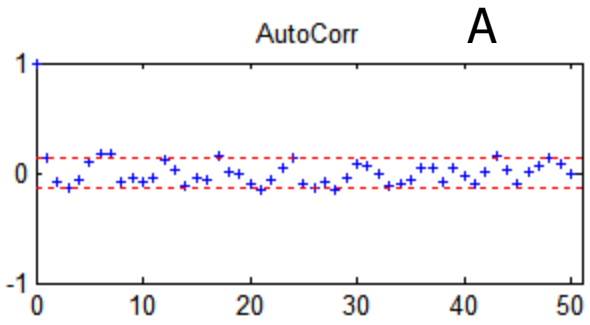


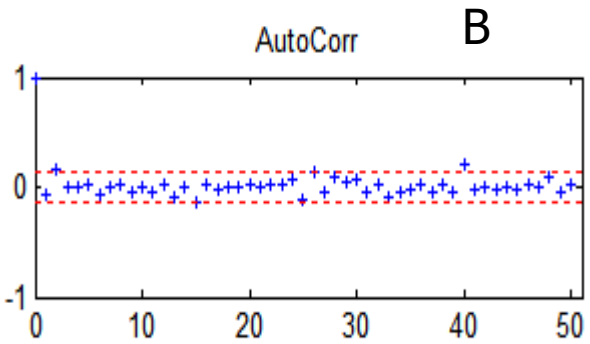
Windows + Linux

All Linux

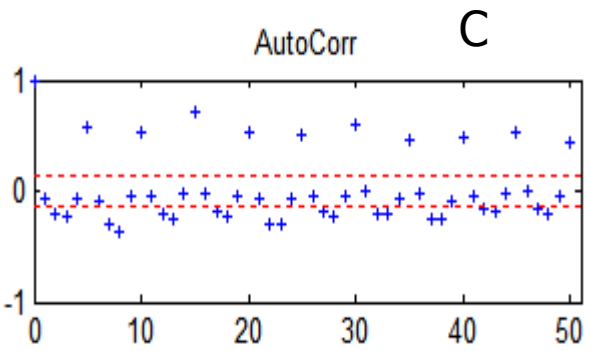
Results II - Autocorrelations

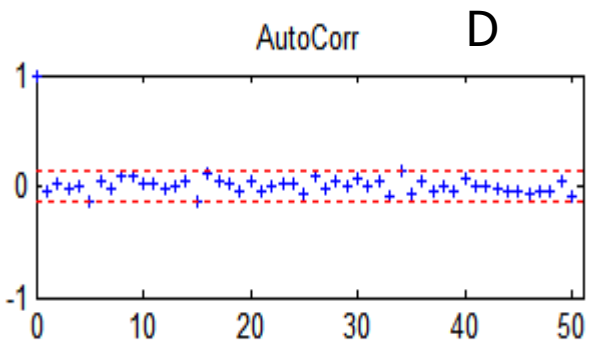
3-machine





2-machine





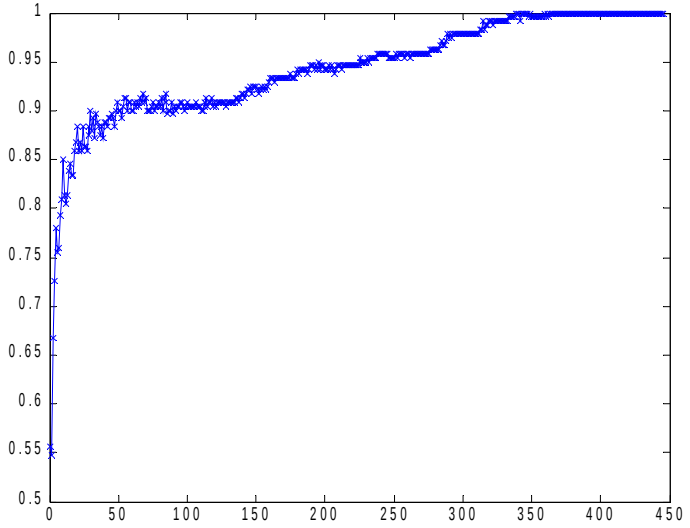
Windows + Linux

All Linux

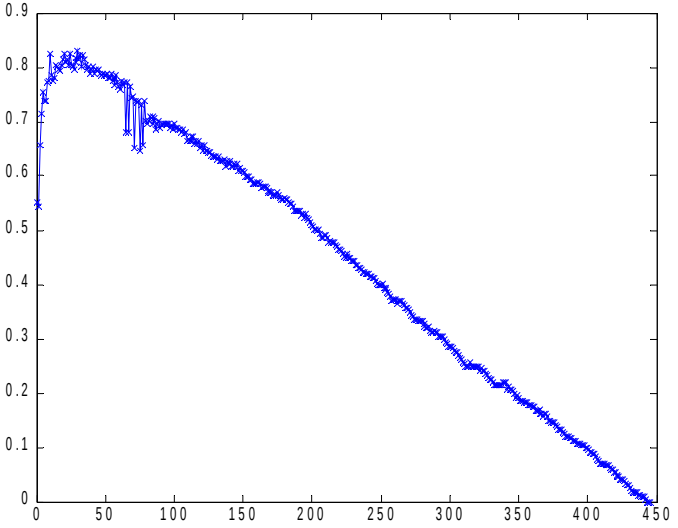
Results III – Sensitivity to N (Run 3 Configuration C)

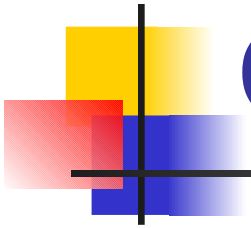
Most accurate N (observation window size) depends on:
Actual conditions AND initial fuzz factor setting
Generator set to produce 241/445 "real" failures
With large N we use initial fuzz factor longer, erroneously reporting fewer "real failures" (when we're missing real problems)

Initial fuzz factor setting = 0 ms
85%-90% accuracy with smaller N



Initial fuzz factor setting = 500 ms
80%+ accuracy with smaller N.

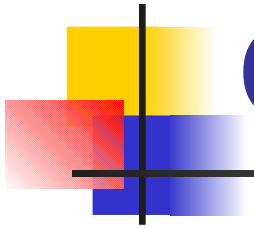




Conclusions

- There is more to our notion of "propagation skew" than network delays. Resource contention at the receiver on certain platforms as seen in configuration C (2-machine Linux + Windows setups) also affects our observations.
- Near optimal settings automatically achieved by managing the tradeoff between larger observation windows and the ability to respond quickly to changes in the environment.
- Feed-forward control useful in building self-regulating systems that rely on temporal event correlation.

Comments, Questions, Queries



Thank you for your time and attention.

Contact: Rean Griffith
rg2023@cs.columbia.edu

Event Package

- Events Represented as Siena Notifications of size ~80 bytes

$$E_1 = \{ FPGenGap = "0" FPResolution = "2000" FPSeqNum = "1" FPStartSeq = "1" FPTest = "FPTest" \}$$
$$E_2 = \{ FPGenGap = "2041" FPResolution = "2000" FPSeqNum = "1" FPStartSeq = "0" FPTest = "FPTest" \}$$