Network Management using Mobile Agents Framework (MAF)

Sproj-0315

Yasir Siddique Sheikh (2003-02-0218)

Ali Bilal Aslam (2003-02-0018)

Ibrar Javed (2003-02-0079)

Shahzad Ismail Mian (2003-02-0188)

Advisor: Dr. Zartash Afzal Uzmi

The Idea!

- ROC; Recovery Oriented Computing

  Joint Project of Berkeley & Stanford University:
  Headed by: Dr. Armando Fox
  http://roc.berkeley.edu

- Autonomic Computing / Self-Healing

  Project of IBM® Watson Research Laboratories, Massachusetts :
  http://www.research.ibm.com

Project Objectives

1.To learn more about the Mobile Agent Protocol developed by IBM Corp., patented as Agent Transfer Protocol (atp://) and try to implement Network Management Functionalities using this efficient protocol.

2. To learn more about the SNMP (Simple Network Management Protocol) and explore its use in the areas of Performance Monitoring and Fault Detection.

3. To exploit the features presented by JAVA programming language, which help in developing Mobile-Agent-based applications. One of the significant features provided by JAVA is that we can serialize our objects while transferring them from one host to another.

4. The most important objective is to research and then try to implement the Algorithms which provide an efficient and authentic way to detect Faults and monitor the performance of a specified network node. What was also important was to be able to devise a framework where by mobile agents could be used to perform the task of monitoring the network.

5. To develop prototype application that would justify our findings and research

Network Management

- Fault Management
- Accounting Management
- Configuration Management
- Performance Management
- Security Management

Conventional Techniques of Network Management

- Centralized; Client Server Architecture

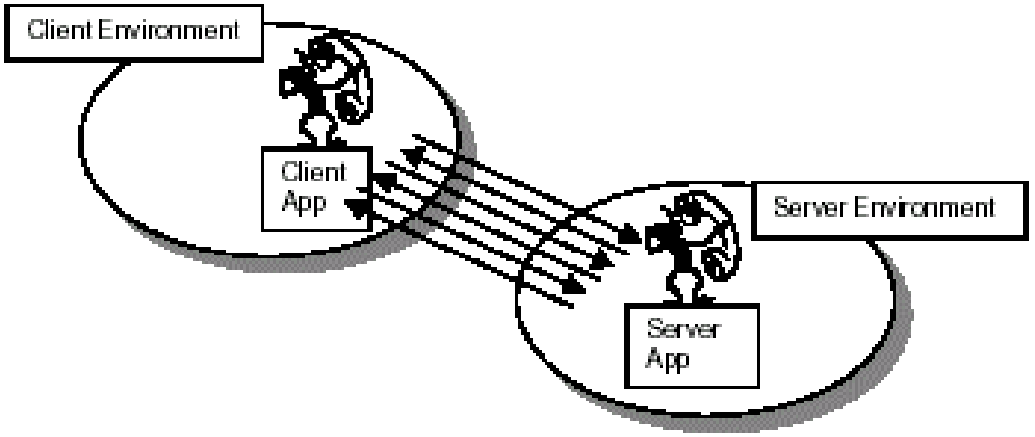


Figure 1: Client/Server architecture

Problems with Client-Server Architecture

- Increased Bandwidth Consumption
- Unreliable network connection
- Rigid Architecture
- Inefficient Load Management

Proposed Architecture for Network Management

- Mobile Agent Frame Work

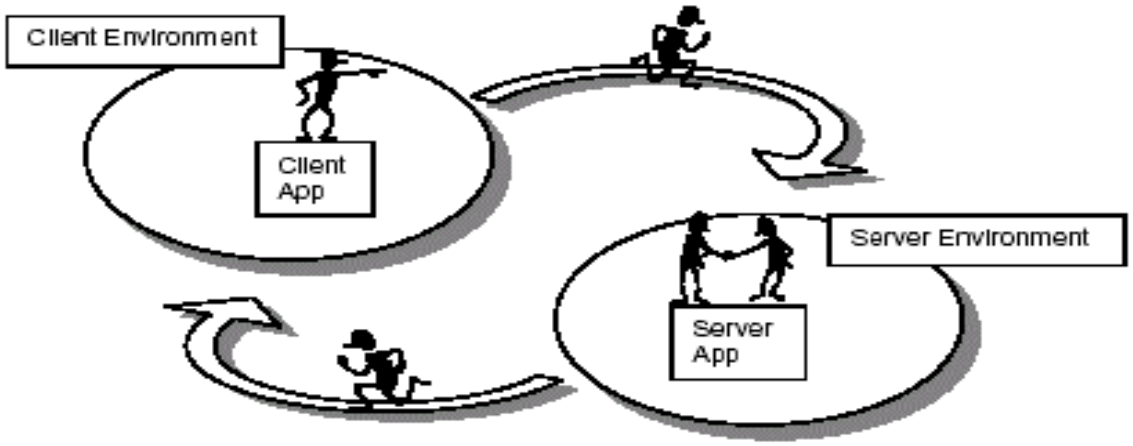


Figure 2: Mobile agent architecture

What is an Agent?

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russel & Norvig 1995)."

What is a Mobile Agent?

An Agent which is able to transport itself from one machine to another

How does Mobile Agent Framework solves the Problems?

- Increased Bandwidth Consumption ⇒ Query / Transaction migration
- Rigid Architecture ⇒ Flexible design
- Unreliable network connection ⇒ Offline Management
- Inefficient Load Management ⇒ Parallel Processing

Development Tools

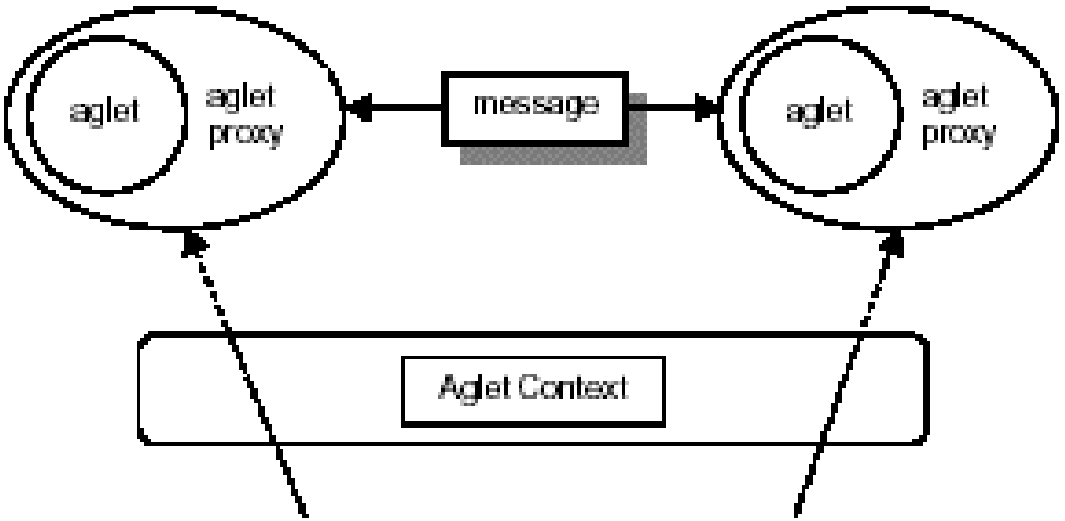
- Java™
- IBM® Aglets

Why JAVA™

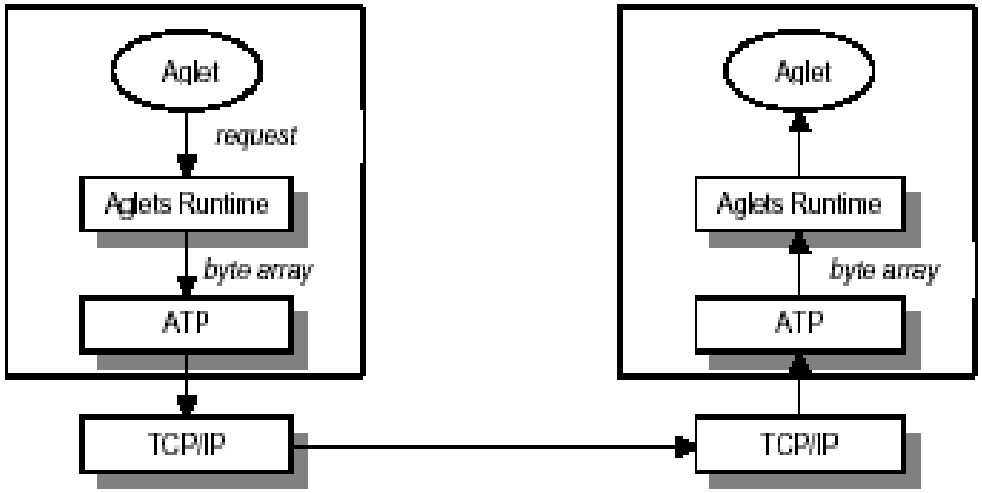
- Object Serialization
- JAVA Security Model (Key Generator)

IBM Aglets

- Aglet Object Model



Aglets System Architecture



Simple Network Management Protocol (SNMP)

- Get
- GetNext
- Set
- MIB Groups
  - 9 Different Groups

SNMP for our Project Purposes

- A Non-Proprietary JAVA API (JMGMT Developed by Mr. Sven Dörr, Germany in 1998)

Integration of SNMP with our framework

- We have embedded the SNMP in our Mobile Agent Framework.
- These mobile agents visit the remote node and obtain the SNMP information from the MIB Database.
- This information is then transported back to the management station

Our Focus…

- Fault Management
- Performance Management

Fault Management

- Deals with detection, isolation and *possible* recovery of faults inherent in a network environment.
- Deals with faults situated inside and outside a node.

Motivation for Fault Management

- Heterogeneous networks and increasing reliance on network resources makes downtime intolerable and costly.
- Hence, fault occurrences need to be minimized.

Proactive vs Reactive Fault Detection

- Proactive schemes are preferred as:
  - Minimizes the downtime and incurred costs due to a fault.
  - Has potential for detecting previously unknown faults.
- A disadvantage is more false alarms.

Network Traffic Based Approach

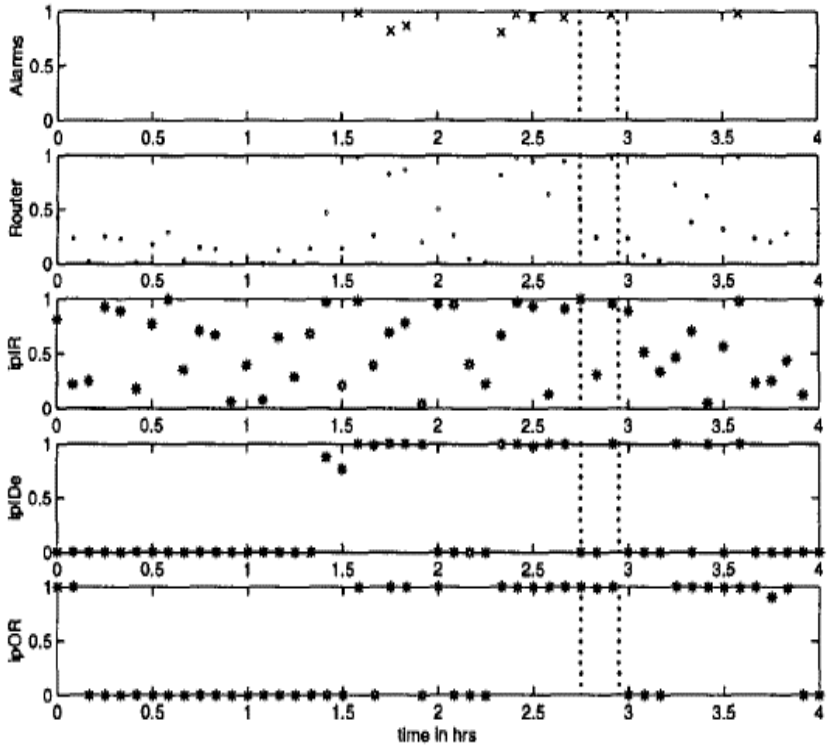
- This is the approach we took.
- Based on an IEEE paper *"Adaptive Thresholding for Proactive Anomaly Detection"* by Dr. Marina Thottan and C. Ji.
- Framework based on predicting faults using network traffic flow changes.

Types of faults detected by this approach:

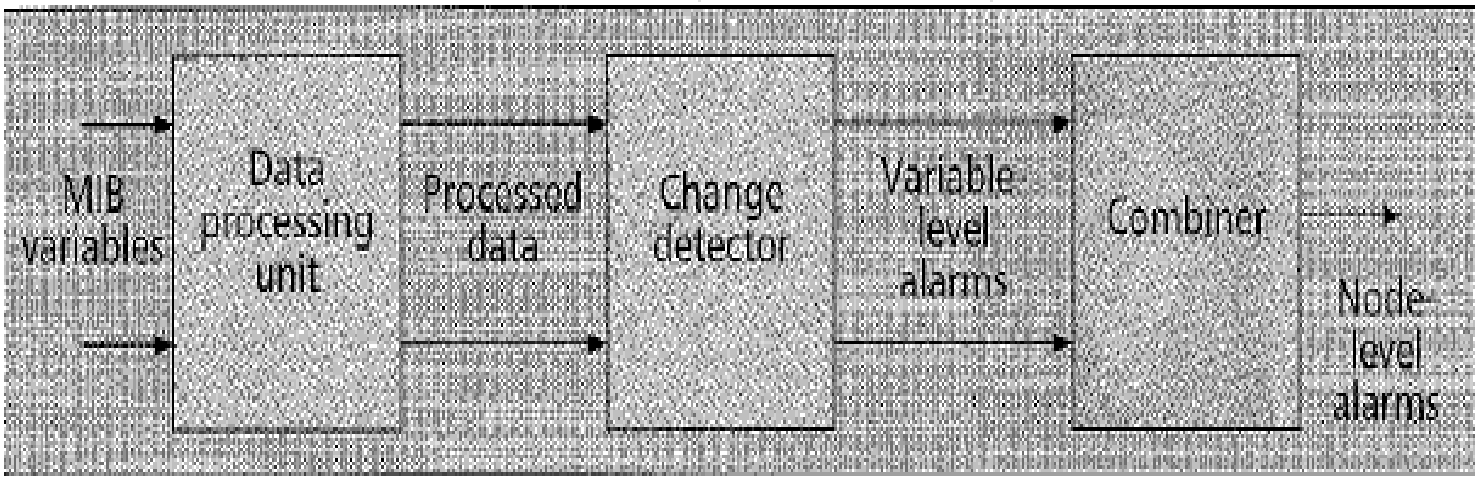
- Performance degradations
- Malfunctioning of network devices
- File server failures
- Cable degradations
- Broadcast storms

Potential Use of Mibs in Fault Detection

- Statistically verified that mibs characteristics change before a network fault occurrence.



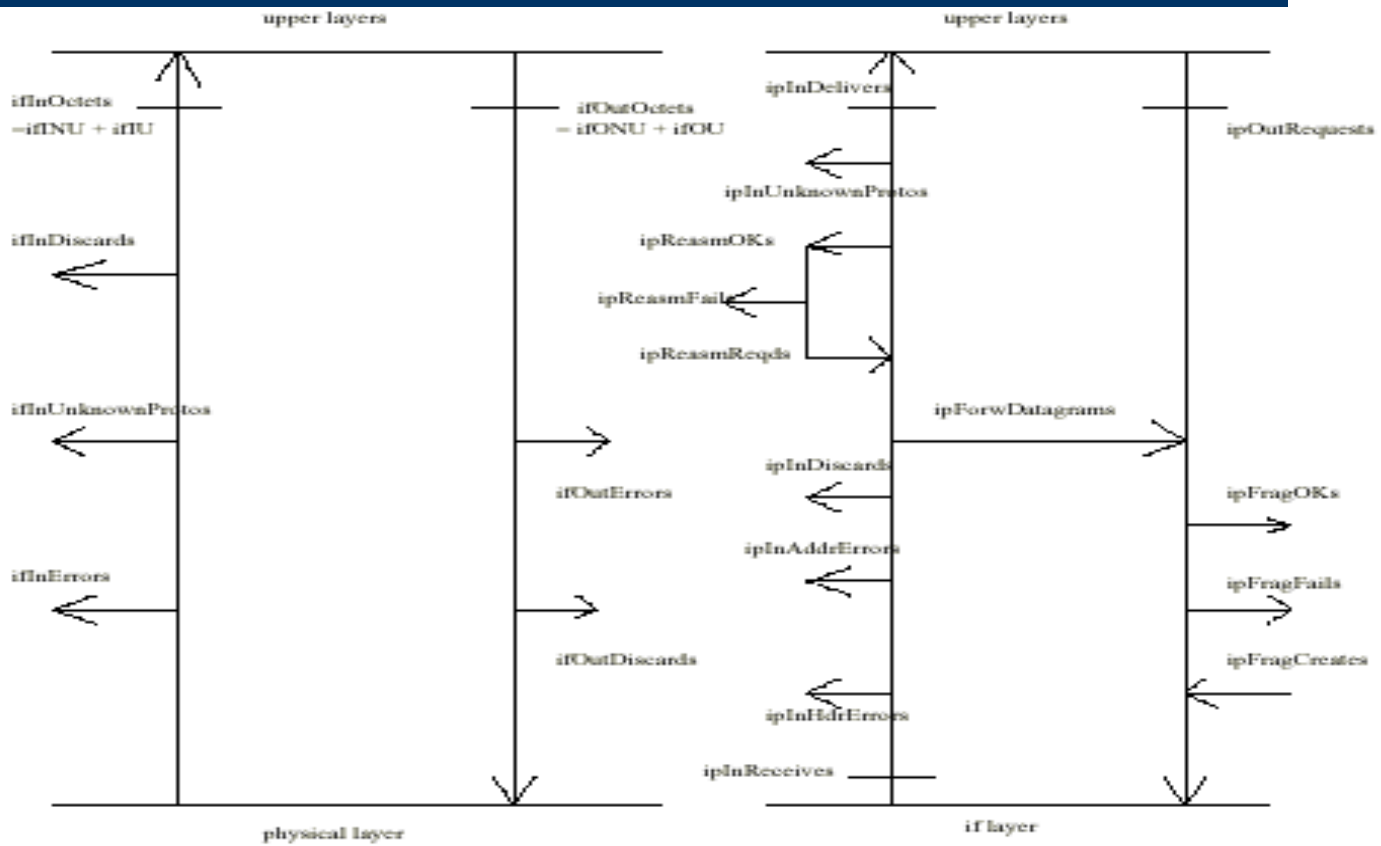
Fault Algorithm: System Model



Choice of Mibs

- Remove variables not relevant to network fault detection.
- Remove Redundancy present in a particular MIB group.

Removing Redundancy?



The Chosen Mibs…

- ifInOctets
- ifOutOctets
- ipForwardDatagram
- ipInReceives
- ipInDelivers
- ipOutRequests

Variable-Level Alarm Generation

- Alarms generated using a change detection algorithm.
- Algorithm divided into two parts:
  - Detect changes within the network traffic
  - Detect whether the change corresponds to abnormal behavior.

How to detect changes in network traffic?

- Use time-series statistical analysis based on generalized likelihood test and a hypothesis test.
- The equation used:

$$-\ln\lambda = \acute{N}_R(\ln\hat{\sigma}_P - \ln\hat{\sigma}_R) + \acute{N}_S(\ln\hat{\sigma}_P - \ln\hat{\sigma}_S)$$

- Change detected if value goes beyond some threshold.

And how to detect abnormal change?

- Use another hypothesis test.
- This time the pooled data set is compared to a normal data set gathered over a 24 hour period.
- This data set represents normal network behavior.

Node-Level Alarms

- A simple strategy used to combine alarms generated by each of six variables.
- Used a weight for a particular variable based on its relevance to fault detection.

MIB variable	avg no of alarms per hour	no: of faults detected
ifIO	0.91	4/9
ifOO	1.14	5/9
ipIR	3.97	7/9
ipFD	3.98	7/9
ipIDe	2.08	7/9
ipOR	1.19	9/9

Performance Management

- Utilization of Network card
- Number of Packets discarded due to resource restriction
- Number of Packets discarded due to errors

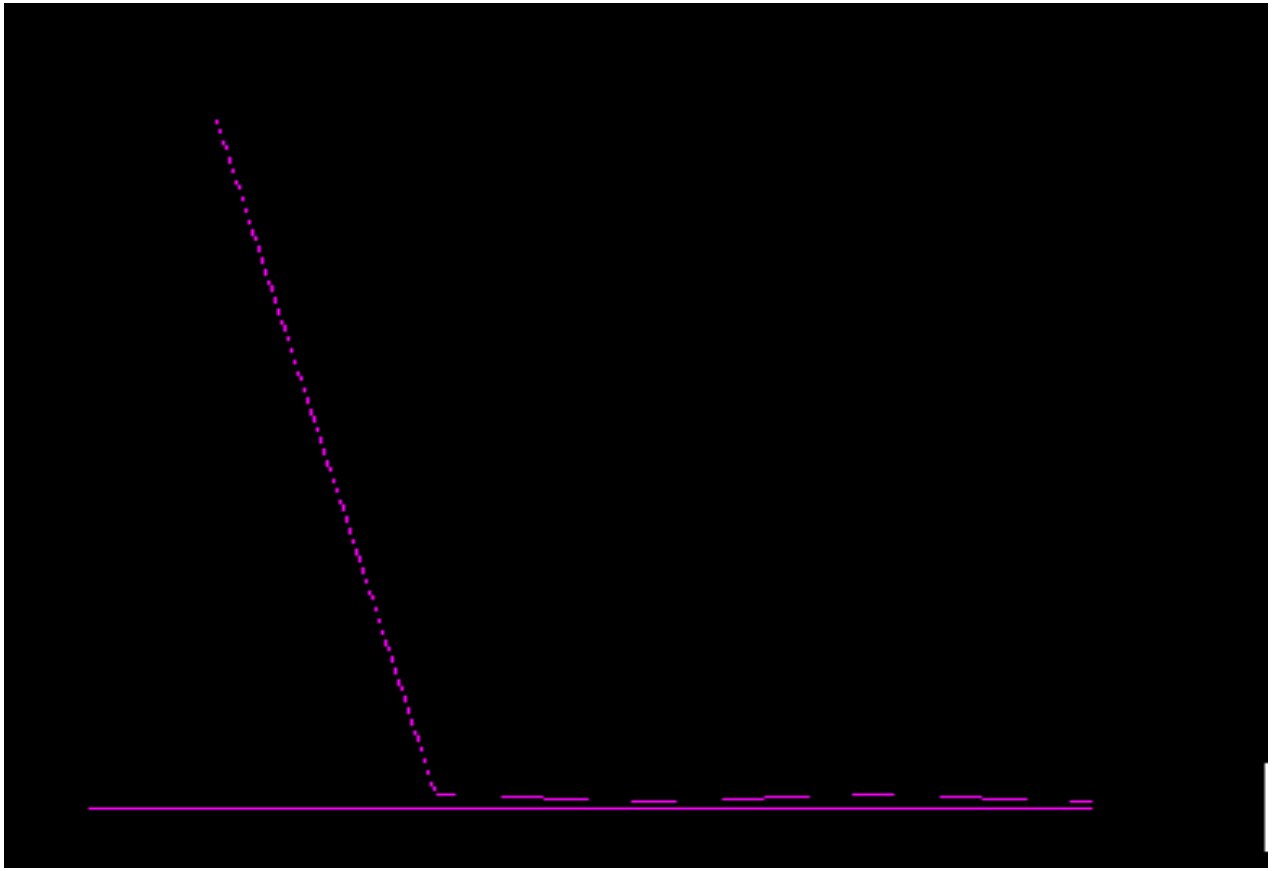
Utilization

- A measure of utilization of Network Interface card (NIC)
- We followed the ground laid by *Iwan Adhicandra* and *Colin Pattinson* in their article *“Performance Evaluation of Network Management Operations “*

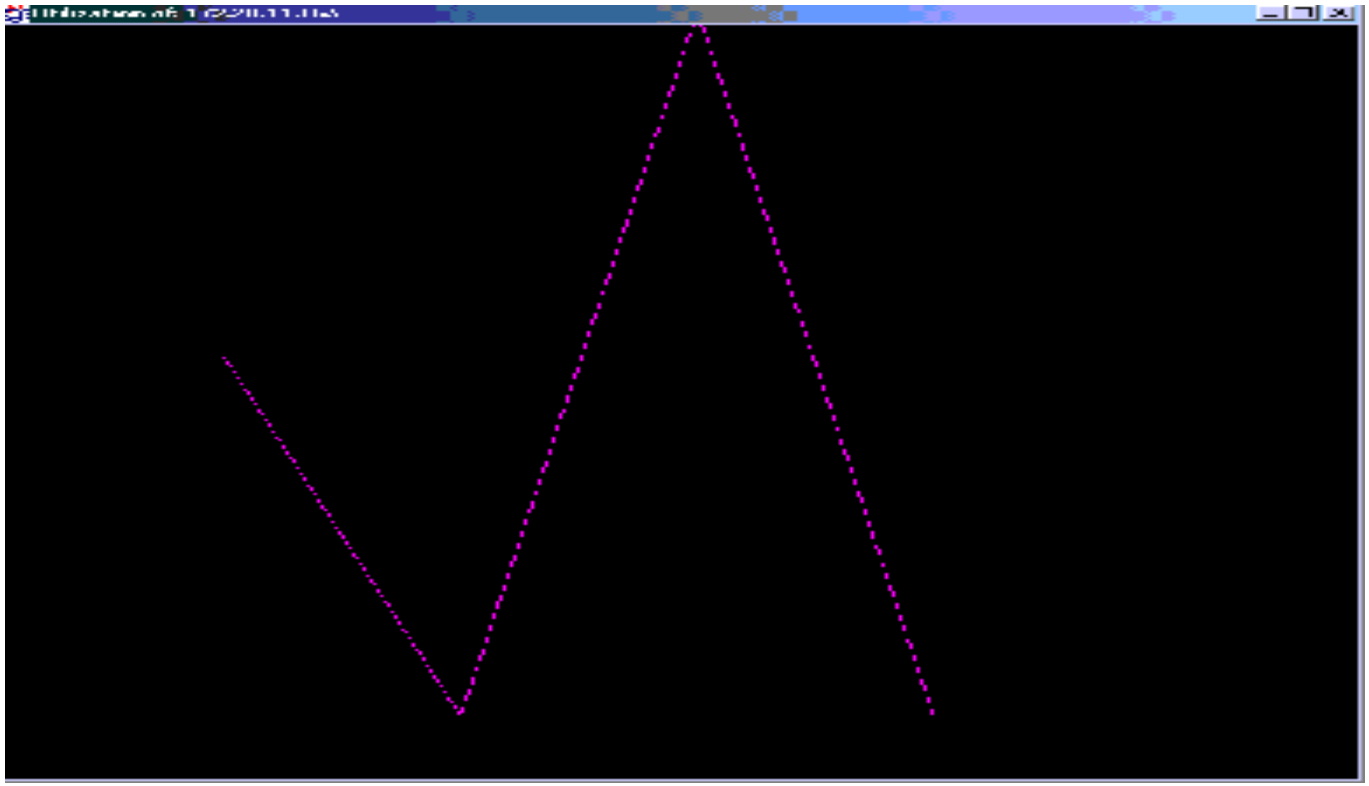
How to calculate utilization

- ifInOctets and ifOutOctets in ifTable in the interface group
- *Total bytes = (ifInOctetsy – ifInOctetsx) + (ifOutOctetsy – ifOutOctetsx)*
- *Total bytes per sec = total bytes / (y – x)*
- *Utilization = (total bytes per sec * 8) / ifSpeed*

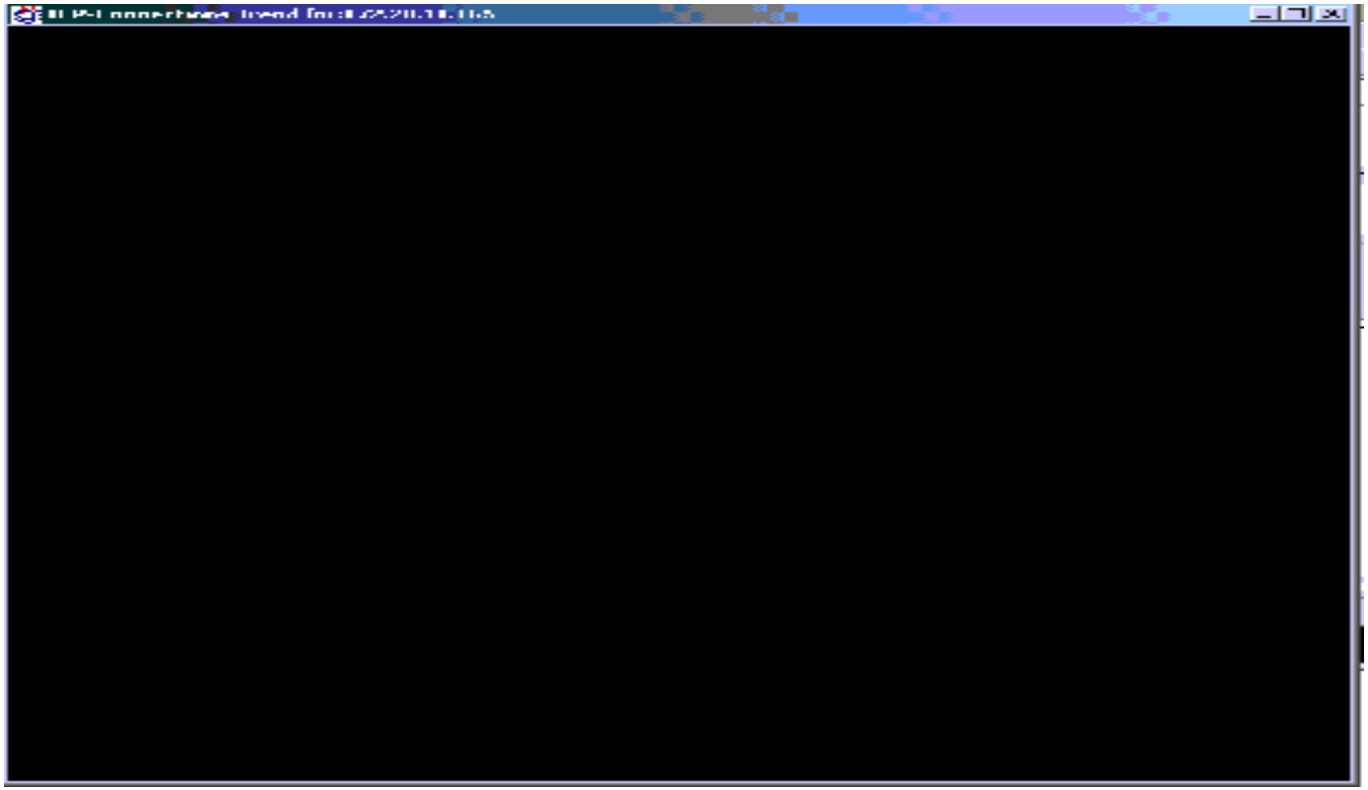
Low utilization



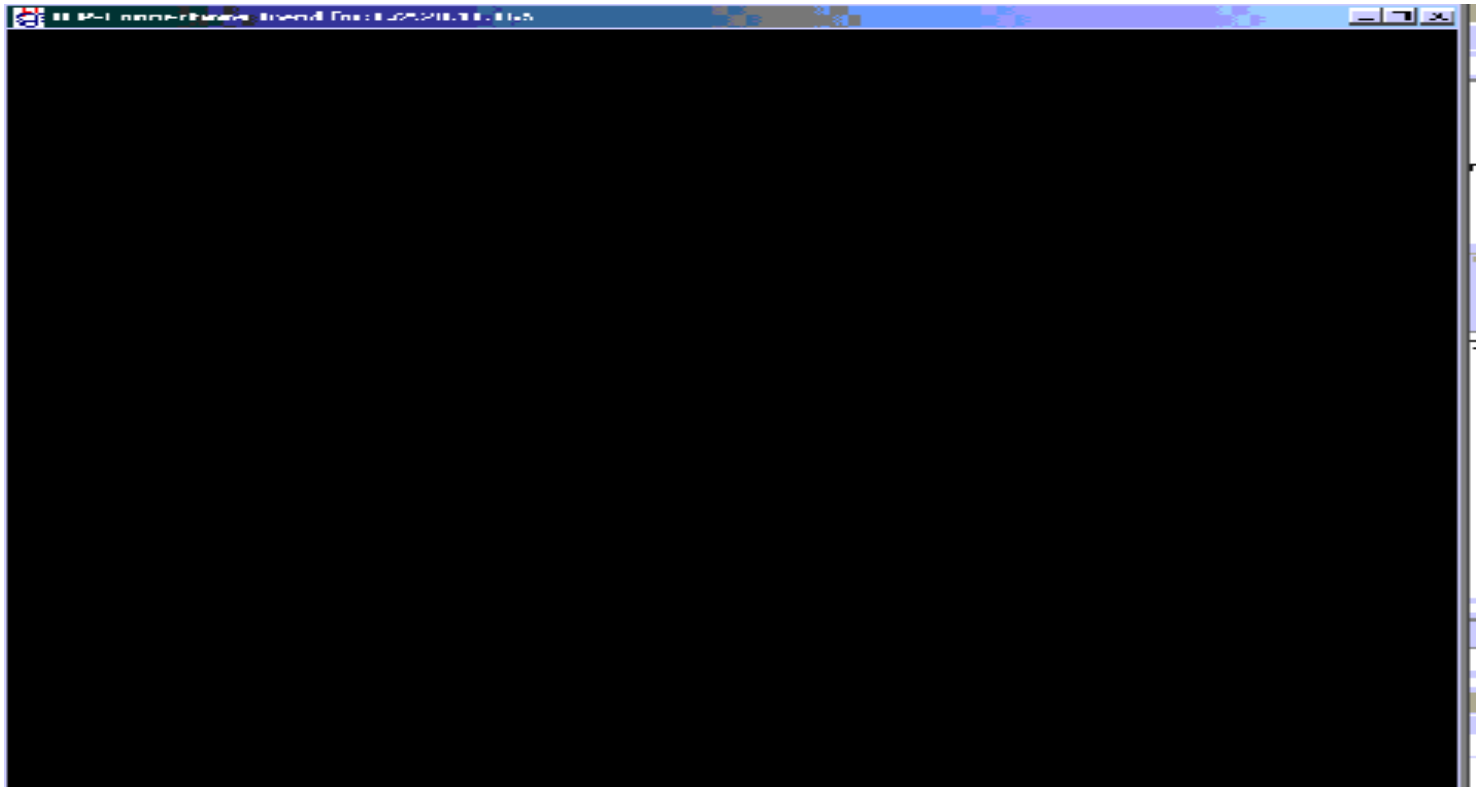
High Utilization



Number of packets dropped due to resource restriction (IfInDiscards)



Number of packets with errors (ifInErrors)



Performance monitoring

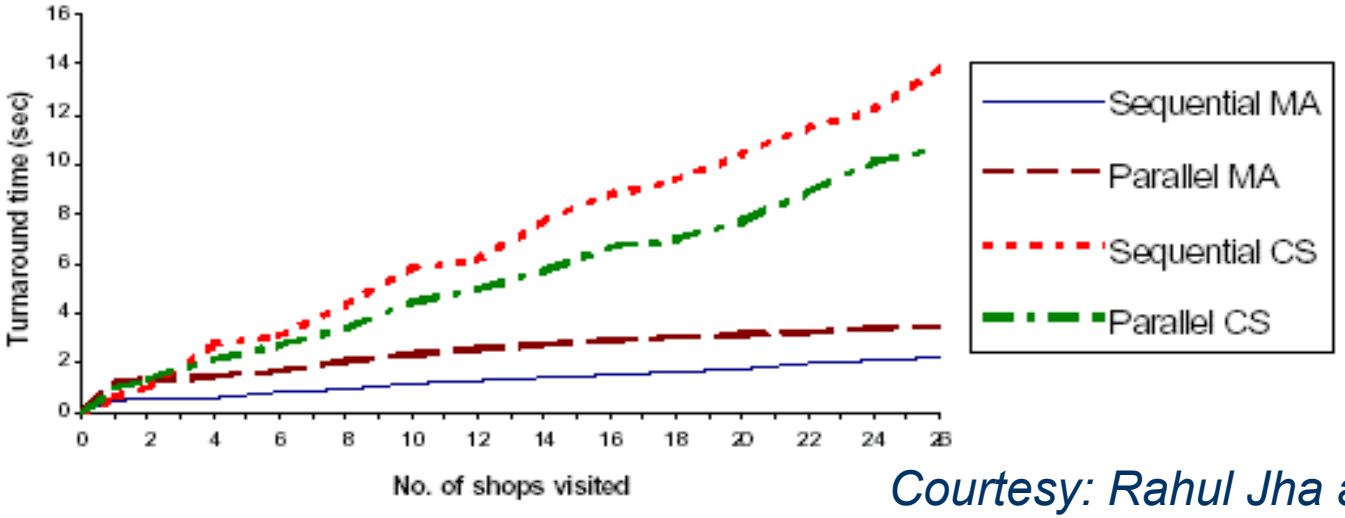
- We have now developed the framework for performance monitoring
- We can now perform analysis based on any combination of MIB variables available such as ICMP messages and packet exchange information

Conclusion

- SNMP despite it's centralized nature can be effectively used with Mobile Agent Framework.
- Java because of its Serializable and Platform independence is the most suitable language for Network Management.

Conclusion…

- Comparison with Conventional Client/Server Model



Courtesy: Rahul Jha and Sridhar Iyer

Future Work

- A better node-level alarm generation scheme using duration filters.
- Cluster based Classification of Specific Network Faults leading to remote recovery computing.
- Extend it to configuration and accounting management
- Authentication is guaranteed by atps:// but confidentiality is an issue

Problems Encountered

- Serialization problem
- Security Configuration of Mobile Agents
- Poor SNMP API JavaDocs
- Fault Algorithm Testing
- Dr. Armando Fox refused to be our external advisor…