DYNAMIC CONFIGURATION MANAGEMENT USING MOBILE INTELLIGENT AGENTS

A Thesis by

Karthik HanumanthaRao

Bachelor of Engineering, Anna University, India, 2005

Submitted to the Department of Electrical and Computer Engineering
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

May 2009

DYNAMIC CONFIGURATION MANAGEMENT USING MOBILE INTELLIGENT AGENTS

The following faculty has examined the final copy of this thesis for form and content, and recommends that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

_____

Ravindra Pendse, Committee Chair

_____

M. Edwin Sawan, Committee Member

_____

Krishna Krishnan, Committee Member

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Dr. Ravi Pendse, for his unconditional and continuous support over the entire course of my master's degree at WSU and for giving me an opportunity to do my Master's thesis under his guidance. He was always there to listen, advice and guide. I also thank him for giving me the opportunity to work in his Tac research lab which helped me to hone my skills in the field of networking.

Besides my advisor, I am thankful to my committee for taking the time to work with me in this regard.

I would also like to thank Mr. Amarnath Jasti, for all his valuable suggestions and asking lot of good questions which helped me to think through the problems.

Finally, I thank my parents and friends for supporting and encouraging me throughout my life to achieve higher goals.

ABSTRACT

With the increasing complexity of Aircraft Data Network the need for configuration management has become a necessity as the tolerance level is absolutely minimal. It is also known that most of the network faults are due to invalid configurations on the network devices. As fault detection plays a vital role in detecting invalid configurations, configuration management forms its basis.

The aim of network configuration management is to supervise the network information so that the changes on networks can be tracked and managed. Also the topology of network can be better understood with each device's configured parameters (interface settings, routing protocols etc).

With the existing web based framework for Aircraft Data Network, we could use it for configuration management by adding a few more modules. The reason for using web based is that it can be controlled remotely and its ease of use. And also our web based architecture uses secure protocols and a centralized database. Mobile agents are used to carry the necessary data to configure the nodes in the network. The suggested framework would reduce the complexity of network configuration as well as improve the performance with reduced network time-delays and information bottlenecks.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

Figure Page

LIST OF FIGURES (continued)

Figure Page

LIST OF ACRONYMS

IP	Internet Protocol
WWW	World Wide Web
ISO	International Organization of Standardization
API	Application Programming Interface
JVM	Java Virtual Machine
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
XML	Extensible Markup Language
JSP	Java Server Pages
ADN	Aircraft Data Networks
SNMP	Simple Network Management Protocol
RMI	Remote Method Invocation
NMS	Network Manger System
MIB	Management Information Base
TCP/IP	Transport Control Protocol
UDP	User Datagram Protocol
OID	Object Identifier
ASN	Abstract Syntax Notation
PDU	Protocol Data Unit
SMI	Structure of Management Information

LIST OF ACRONYMS (Continued)

BER	Basic Encoding Rules
ISM	Intelligent System Modeler
DB	Database
IPV4	Internet Protocol Version 4
IPV6	Internet Protocol Version 6

LIST OF SYMBOLS

$\sum$	Summation
λ	Lambda
Γ	Gamma
BW_{util}	Bandwidth Utilization
$BW_{util\ ipv4}$	Bandwidth Utilization for IpV4
$BW_{util\ ipv6}$	Bandwidth Utilization for IpV6
$P_{c\ ipv4}$	Length of bulk transfer packet for IpV4
$P_{c\ ipv6}$	Length of bulk transfer packet for IpV6
P_{ipv4}	Size of IpV4 packet
P_{ipv6}	Size of IpV6 packet
N_e	Number of elements to be monitored
N_o	Number of OIDs per element

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

The success of the Internet and its increasing popularity has significantly strengthened the position of aviation industry. Many modern network applications are outcome of this industry. Network management plays a key role in ensuring the efficient operation of such a network infrastructure. In this modern era where only change is constant, network management has to address a number of requirements for dynamic, reliable and continuous operation. In modern aircraft network the push for greater economy has resulted in the reduction in size of the flight crew and increased the number of networks by many times. This increase in number of IP networks has also increased the complexity of the network and makes the network administrators life miserable. End-to-end network requirements can be on connectivity, security and fault-tolerance.

Web-Based network management uses the World Wide Web to suffice the problems of network administrators. The most important advantage of using Web Based is that it makes manageability simpler. As per ISO management framework, the configuration management system should be able to control any managed object, to collect data from managed objects and also be able to provide management information to network manager.

1.2 Network Configuration Management

Network configuration management system should be able to perform simple network operations like shut a network resource, modify the configuration of any network element, add new or remove any existing network element to and from the network.

1.3 Research Motivation

With increasing IP networks many sophisticated network management applications were developed to suffice the needs of configuration management. The initial protocol based approach had a lot of limitations which is highlighted in [36]. Because of its centralized and static nature it did not make much success. As an alternative, distributed framework approaches were proposed. Even this approach did not make much success mainly because of lack of efficient programmability. This led to the evolution of mobile agent technology that addressed the issues of both protocol based and distributed approaches.

1.4 Objectives

The thesis aims to explicate the potential impact of mobile agents towards network configuration management aided with the proposed architecture. The main objectives are:

1. Network Configuration management architecture proposal
2. Identification of agent migration
3. Evaluation of mobile agents in the context of configuration management.
4. Mobile agent proposal suited for our architecture.

1.5 Problem Description

In this thesis, the author proposes an Intelligent System Modeler that is used to automate the network configuration management tasks. This Intelligent System Modeler takes as input a set of network components and computes as output, component configurations satisfying those requirements. The output from modeler is realized as mobile code. Mobile agent is a process that can transport its state from one environment to another with its data intact and still be able to perform appropriately in the new environment. It provides user interfaces using HTML, JSP, XML and other features.

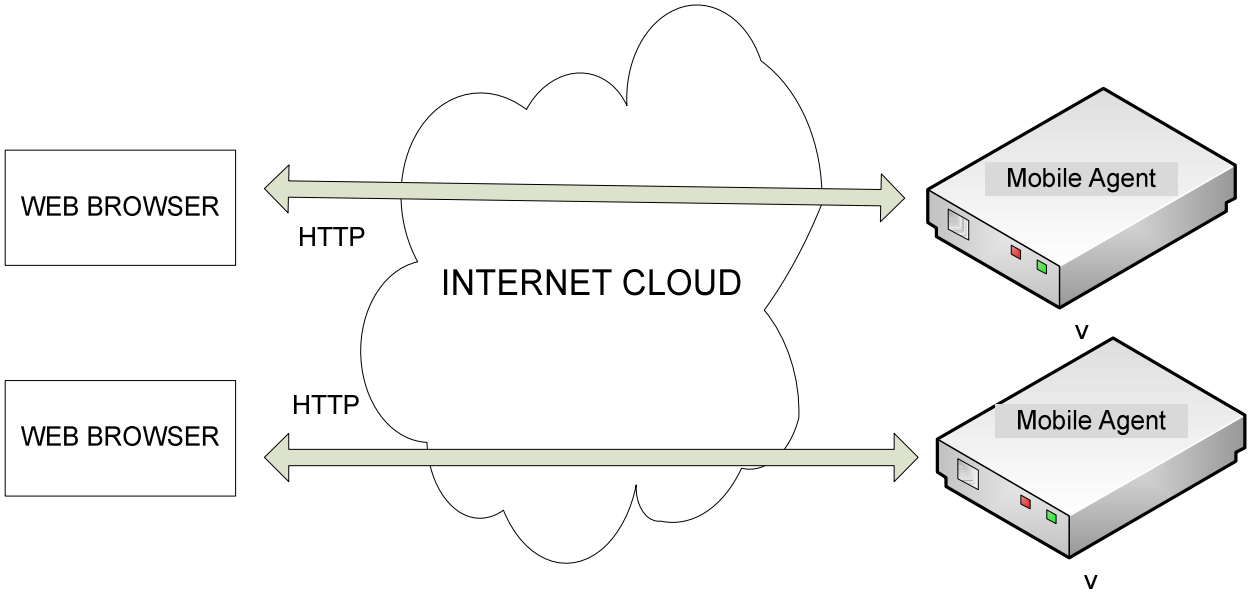

Figure 1 Web Based Network Management architecture

Figure 1 shows web based management architecture. There are plenty of advantages of using Mobile agents ie; low cost, easy to use, easy maintenance. Mobile Agents has low resource utilization, high reliability, portability and high security.

In addition, Configuration management system should also be able to provide information of network elements, all through user-friendly GUI's for network manager to access and control the whole network.

1.6 Thesis Roadmap

The rest of the document is organized as follows:

Chapter 2 describes the background topics associated with this interdisciplinary thesis. These include the protocols used, mobile agents with software mobility and their approach for network management.

Chapter 3 discusses the detailed description of the proposed architecture.

Chapter 4 deals with the mathematical analytical model for the proposed architecture.

Chapter 5 deals with Performance measurement with simulations and results.

Finally, Chapter 6 draws conclusion and offers suggestion for future work.

CHAPTER 2

LITERATURE SURVEY

This chapter provides a background of various topics in configuration management and software agents.

2.1 Network Management

Network management is defined as a concept that uses various tools, techniques that assist in managing different devices or networks. Network management is classified into five different areas by ISO [42] (International organization for standardization).

2.1.1 Fault Management

The objective of fault management is to ascertain, log and report network or system users of problems. In many circumstances, recess of any kind is unacceptable.

2.1.2 Configuration Management

The objective of network configuration management is to administer the network information so that the changes in networks can be recorded and governed.

2.1.3 Accounting Management

The goal of accounting management is to ensure that computing and network resources are used fairly by all groups who access them. Most of the network problems can be resolved by distributing the network resources.

2.1.4 Performance Management

The goal of performance management is to measure and report on various aspects of network or system performance.

2.1.5 Security Management

The goal of security management is twofold. First, we wish to control access to some resources, such as a network and its hosts. Second, we wish to help detect and prevent attacks that can compromise networks and hosts. Attacks against networks and hosts can lead to denial of service and sometimes even allow hackers to gain access to vital systems that contain accounting, payroll, and other valuable information.

2.2 Configuration management and its importance

Due to the increasing complexity of today's IP network a very high level of automation of configuration management has become a necessity. Automatic network configuration management significantly eliminates misconfiguration of networks which is the root cause of many network faults. Configuration tasks require several high-level management operations such as download, activation, rollback and restoration.

2.3 Intelligent network monitoring system

Authors of [1] have already proposed and implemented a monitoring tool that would provide a live status of the network to the crew and the ground station. As mentioned in [1] it is evident that some of the available market tools for network management are not custom tuned for Aircraft Data Networks. Therefore a need to develop a management tool with wide range of statistical information has become a prime importance.

From the complex network design of ADN [2] it is paramount to have a configuration management that complements the network management tool. To fully understand the importance of configuration management it is necessary to know the relationship of various subsystems [3]. The operations add, delete and modify cause a change in states of the subsystem. The change in state of one element due to any one of the operations such as add, delete or modify

triggers a change in state of the neighboring nodes or elements. Therefore it is very important to track the states of the neighboring nodes as well or of the subsystem as a whole.

The XML based architecture proposed by authors of [37] is aimed at providing the following objectives:

- Use RPC based mechanism for communication between manager and an agent.
- Uses TCP as transport layer protocol

The use of TCP would not suffice our needs as the operations of configuration management would be one time and would not affect the performance of the system.

Many self-healing systems can be constructed by carefully integrating the simpler ones. Authors of [38] outline an approach called Service Grammar to self manage the network.

According to their algorithm hierarchical synthesis and analysis of system networks can greatly accelerate the configuration method.

2.4 Technologies for Network Management

There are different protocols available for network management namely SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol). SNMP involves a management station for interaction with agents in the network elements. Remote Monitoring (RMON) is a way to decentralize management tasks.
CMIP is much more sophisticated and require greater amount of computational resources. With SNMP it is much easy to efficiently program than CMIP.

2.5 SNMP

SNMP is a set of operations that gives administrators the capability to modify the state of SNMP enabled devices. It can shut down an interface or check the operating speed of SNMP devices.

2.5.1 SNMP Entities

There are two kinds of entities:

- Managers
- Agents

2.5.1.1 Managers

Managers are otherwise known as Network Management Stations (NMSs). The responsibility of a network management station is to poll and receive trap from other agents in the network. When the Network Management Station actively sends queries to a device that is being managed it is called Polls. If the agent actively sends information about the device being monitored to Network Management Station it is called as traps.

2.5.1.2 Agents

Agents are software code that is run on the device to be managed. By keeping a record of several operational aspects of the device, the agent will provide management information to the NMS.

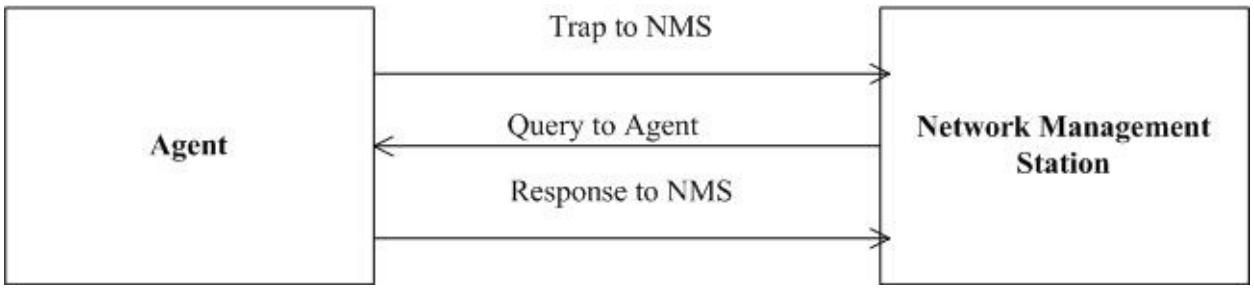

Figure 2 Correlation between the NMS and an agent

2.6 Survey of Mobile Code Technologies

In Agent TCL, the executing unit is implemented by a unix process. The executing units can move to another environment by forking a new process at the remote environment. This way

variables or parameters can be migrated to remote environment. It provides strong support for mobility.

Java has attracted lot of attention because of its portability, easy-to-implement and security features. Java is distributed, which means it can involve several systems on a network to work together. Also keeping in mind the security and reliability, java clearly outshines other programming languages. Also its integration with World Wide Web technology has made it more successful.

In Script MIBS, the management functionalities are installed in network nodes in the form of executables. It uses two technologies namely Push model and pull model. In Push model the manager pushes the script to the distributed manager. In Pull model the manager has to inform the distributed manager of the location of a script.

2.7 Design Issues

There are several models to implement the network configuration scheme.

2.7.1 Client-Server

In this model, C_a (Client A) requests services to S_a (Server A). The server executes the service and result is sent back to client. The client-server does not scale very with increase in IP networks. Also the performance of such models is very poor. This is shown in [6].

2.7.2 Code on Demand

In this model, component A has all of its resources it needs but does not have the logic to manipulate. So A interacts with B to obtain the logic and then A executes it. In this architecture there needs to be a repository with scripts and appropriate scripts are sent to the requested node.

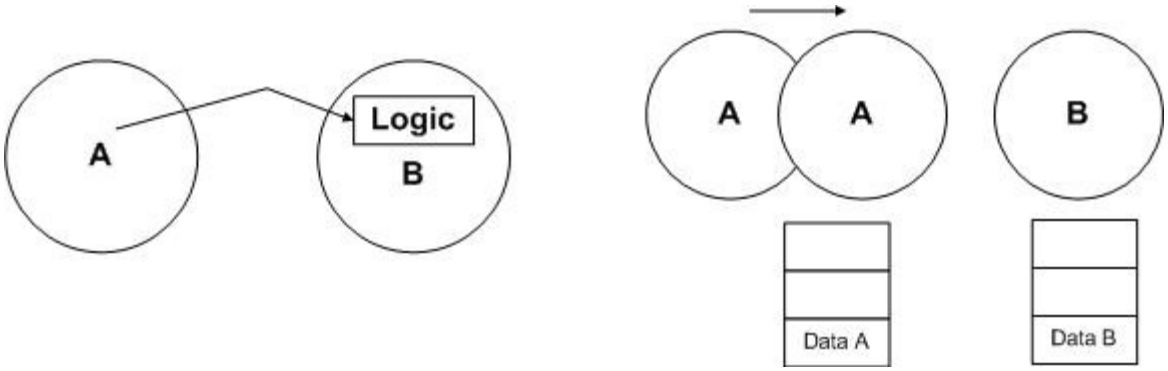

Figure 3 Mobile Agent (a) Code on Demand (b) Mobile Agent

2.7.3 Mobile Agent

In this model, component A has the logic to perform but only with partial resources. So A migrates to B along with its logic and completes the service with resources available there. With Mobile Agents considerable overhead losses can be avoided. Mobile agents co-operate with other agents and act on its own to accomplish the required task. Authors of [39] present architecture with a plug-n-play mechanism that would allow the agents to accomplish complex tasks.

2.8 Need for Java Web Based

With the need to monitor and control the network over a wide geographic the use of web interface aptly suits the requirements [4]. The web model and Java Servlets go hand in hand reducing the complexity, improving security, adding flexibility and lowering the cost. It not only enhances the configuration management as a whole but also makes scalability easier.

2.9 Secure Mobile Agents & its management

Mobile agents are agents that travel across the network and then execute on the remote machine that provide agent host capabilities. The transfer of agents could be done through TCP/IP or higher level such as Remote Method Invocation, Simple Mail Transfer Protocol or

Hyper Text Transfer Protocol each having its advantages and disadvantages [5]. Transfers of agents over network always involve security threats. Some of the vulnerable threats such as masquerading, denial of service, unauthorized authority, confidentiality etc [5] are explored over different models of agent communication.

CHAPTER 3

NETWORK CONFIGURATION MANAGEMENT ARCHITECTURE

Network configuration requirements can be on connectivity, performance and fault-tolerance. With these factors influencing the configuration management, the requirement could be a new one or a modification to the existing network. Our research work focuses on these two areas of configuration management.

3.1 New Requirement

In case of new requirement, the management application is provided with the requirement task. The management application is interfaced with Intelligent System Modeler which computes the output and mobile agents are used to carry out the configuration tasks.

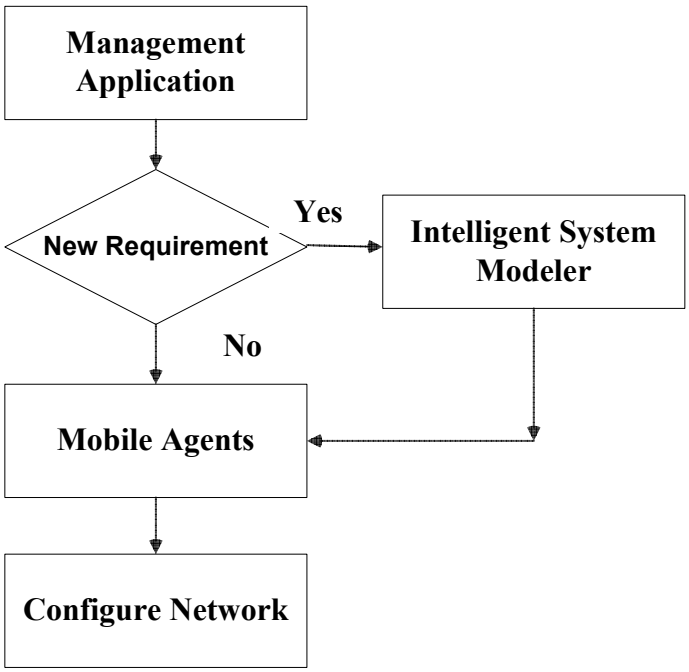

Figure 4 Configuration Management architecture flowchart

The Intelligent System Modeler has been inspired by the algorithm Alloy [4]. Alloy is a math based model but it can be interpreted in Object oriented language easily. Given a requirement it can find object instances which when put together constitute a model. Alloy first takes the requirement as input, computes a formula and then uses a satisfiability criterion [5] to check if the formula is satisfiable.

3.2 Architecture

Network configuration requirements can be on connectivity, performance and fault-tolerance. Automation of network configuration management requires a precisely defined requirement. The intelligent System Modeler takes as input a set of network components and computes as output, component configurations satisfying those requirements. The task is accomplished as follows:

3.2.1 Configuration Construction

Given a set of components with requirement R, the intelligent System Modeler produces the output O.

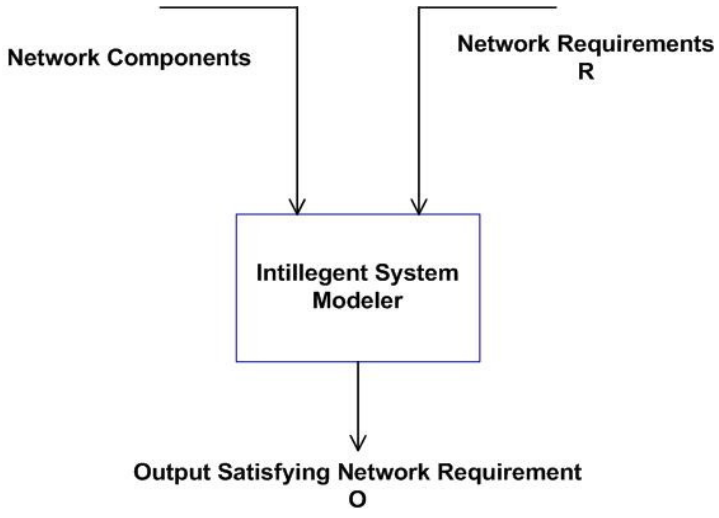

Figure 5 Intelligent System Modeler

- Problem Breakdown: In order to satisfy a requirement, provide a set of elements to the Intelligent System Modeler.
- Addition of new components: Add a new element to the already existing requirement and give to the Intelligent System Modeler and take its output.
- Error Detection: In order to detect errors in configuration, add a few constraints to the requirement and give it to the Intelligent System Modeler, if it cannot produce a solution, an error is detected.

The output(s) obtained from the Intelligent System Modeler may not be efficient, so we add in more constraints to obtain optimal output(s). Although in worst case the solution to the scenario is NP-complete, adding constraints or precise definition of the requirement can lead to efficient solutions in polynomial time [23]. Thus our algorithm helps in identifying if the goals are achievable. This is done by adding a number of constraints to the requirement which in turn makes the domain set smaller in size. With lesser domain set more effective solutions are found in computable time. Adding constraints reduces the solution set but we still need to narrow down to a few.

So we define an action that is associated for goal. Also by associating a warning level for each action we would be able to achieve the goal with minimal disturbance to the network.

3.2.2 Example

For example, the requirement is to establish VPN tunnel between two hosts say flight crew and ground station. The communication is intended to be private, so one way to realize this is by establishing IPSec tunnels. To realize this design the following configuration parameters are necessary:

3.2.2.1 Configuration Parameters

Addressing: Router interface address, subnets.

IPSec: Tunnel end points, encryption algorithms, key scheme

Routing Protocol: OSPF, RIP, IGRP, EIGRP

Firewall Policies: Policies at each site

3.2.2.2 Errors

Errors that can arise are as follows but are not limited to:

1. Duplicate IP addresses may be set or different subnets may be set on same interface.

2. Tunnels may be setup incorrectly, mismatch in keys, hash algorithms.

3. Routing protocol may be incorrectly set: For example, OSPF may not be enabled on required interfaces or area mismatch or hello timer values could mismatch.

4. Firewall policies could block the IP's.

Router Interface Requirements:

1. Each spoke router has internal & external interfaces

Sub-netting Requirements:

1. Internal interfaces are on internal subnets

2. External interfaces are on external subnets

3. No two routers share a subnet

Routing Requirements:

1. EIGRP is enabled on internal interfaces

2. OSPF is enabled on external interfaces

Firewall & IPSec Policy:

1. IPSec is enabled on external interfaces

2. Policies are placed to allow required IP's to communicate with other end of tunnel

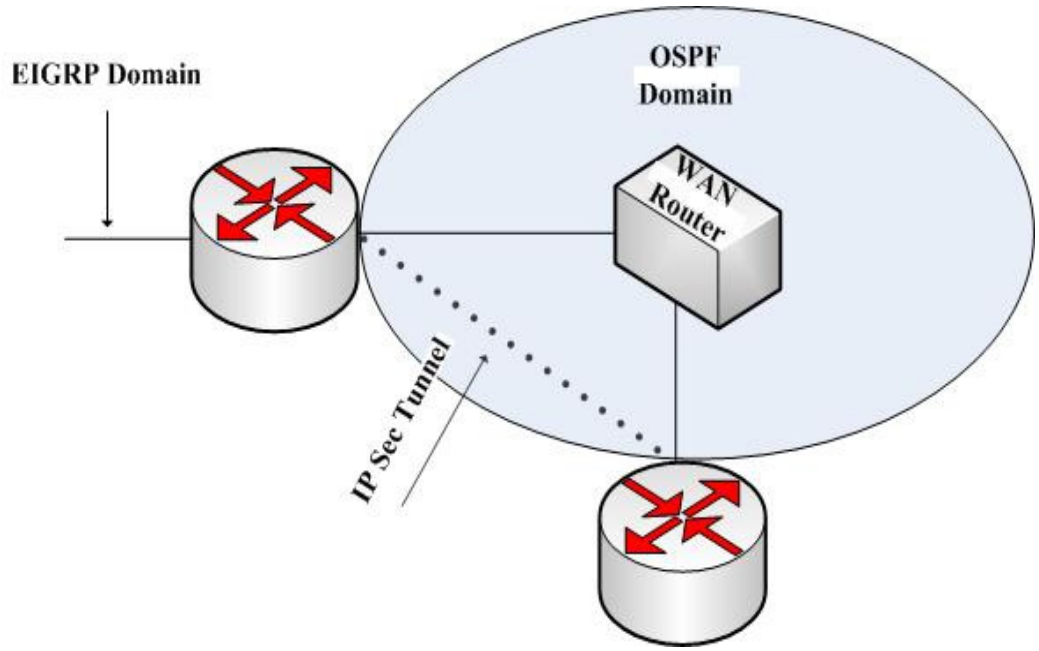

Figure 6 Configuration Construction

The output from the Intelligent System Modeler is then fully realized in scripts or programs. Mobile agents are then used to carry the logic and execute at the remote network to make necessary configuration changes.

Figure 4 shows the model of our architecture. Intelligent System Modeler takes input in the form of requirements and produces an output. The output is then realized into code and sent to appropriate nodes to satisfy the requirement.

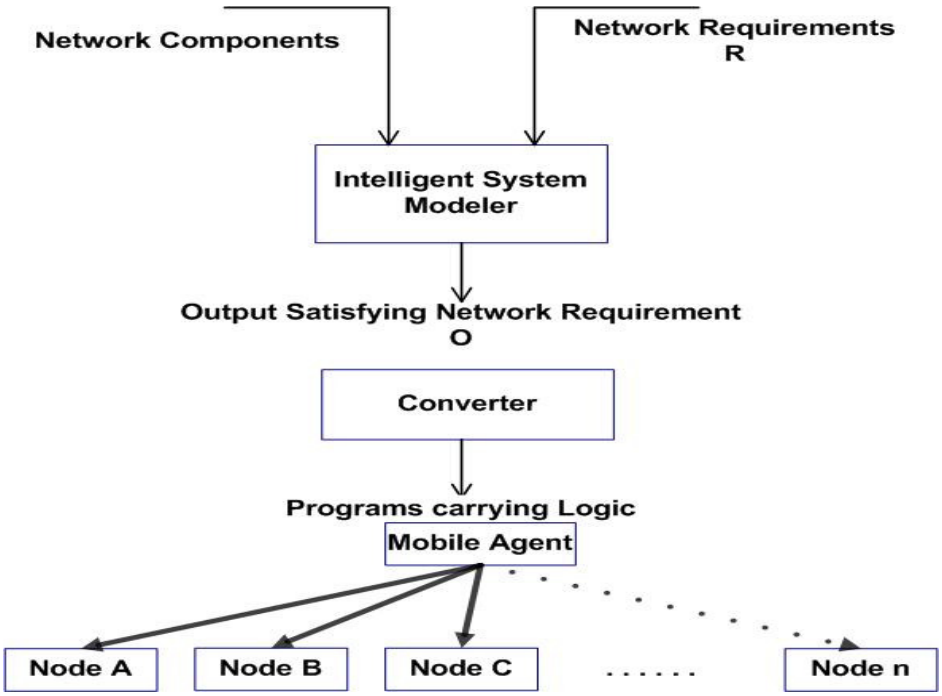

Figure 7 Conceptual Configuration Architecture

3.3 Modification to existing network

Configuration change requirements could also arise on fly due to faults in the network. In order to accomplish such requirements we look at various available architectures and propose one that is suited for distributed architecture.

The traditional configuration management is based on client/server architecture. The management application requests the server through its interface methods to change the state of the application by making changes to one or more network nodes. The server in turn responds or signals appropriate commands to make the necessary changes to network nodes and notifies the application of the result.

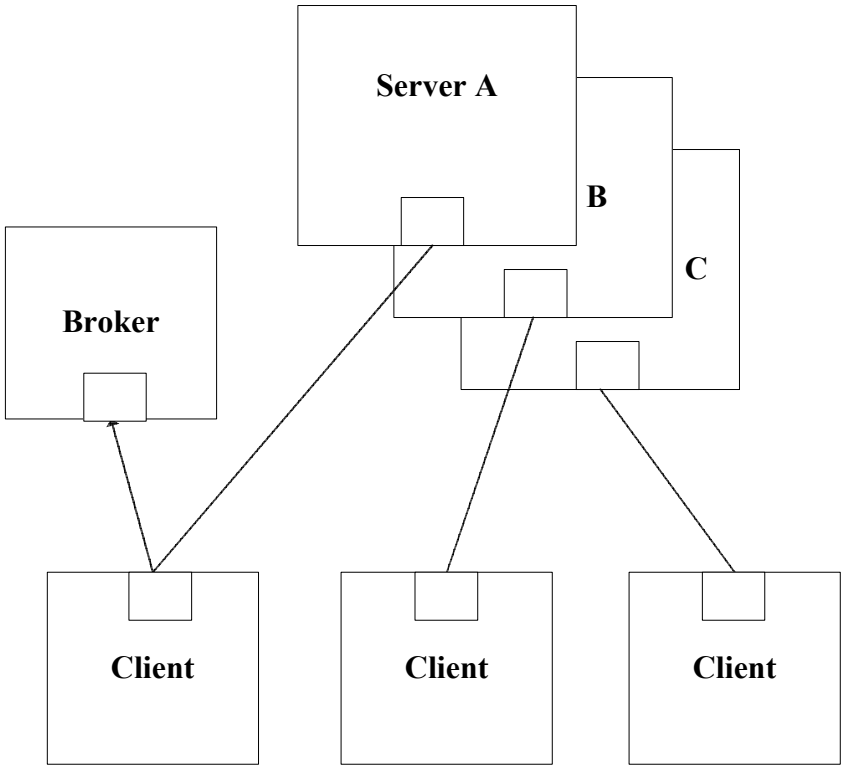

Figure 8 Client Server Architecture

The above figure shows client-server architecture which has several servers replicated. When the broker receives the request from the client to modify the configurations, it binds the client with appropriate server. The client then sends all requirements to the server. The server processes them all and sends the results back to the client.

This kind of architecture has a lot of disadvantages when it is used on large complex distributed networks. With all of the traffic destined to the servers it could cause poor response times and overloaded traffic on a single network segment.

Next we look at policy based configuration management using MIB scripts.

3.4 Script MIB for Configuration Management

MIB's provide a way to define managed objects and their behaviour. The Management Information Base is a database of managed objects that the agent tracks and can be accessed by the NMS to obtain statistical information.

SNMP uses UDP as the transport protocol. As UDP is unreliable, it is up to the application to determine lost datagrams and retransmit if required which is accomplished with a simple timeout. When a UDP connection is established with an agent the NMS waits for a response for a certain period of time. It is possible to configure the number of times the NMS has to retransmit a packet.

3.4.1 Community Names

Community names are basically passwords; and hence there is no actual difference between them. There are three types of community strings.

- **Read-only:** The read-only community string is used to read data and does not allow the user to modify the data.
- **Read-write:** With the help of the read-write community string the user is permitted to read and modify the data values.
- **Trap:** The trap community string is used to receive asynchronous notifications (traps) from the agent.

3.4.2 Managed Objects

Managed objects are storehouse for information. It can be classified into three attributes:

- **Name:** The name also referred as an object identifier (OID), uniquely defines a managed object.
- **Type and syntax:** It specifies the way in which the manager and agent communicate with each other. The advantage of ASN is that it does not have to worry about byte-ordering as the notation is machine independent.
- **Encoding:** Basic Encoding Rules BER is a protocol that specifies the encoding and decoding rules for transmission over transport medium. With the help of BER the managed object is encoded into a string of octets.

3.4.3 Naming OIDs

Managed objects are hierarchical, organized in a tree-like fashion which forms the basis for SNMP's naming scheme. An object ID is a collection of integers based on the nodes, separated by periods (.) **Figure 9** shows the top few levels of this tree.

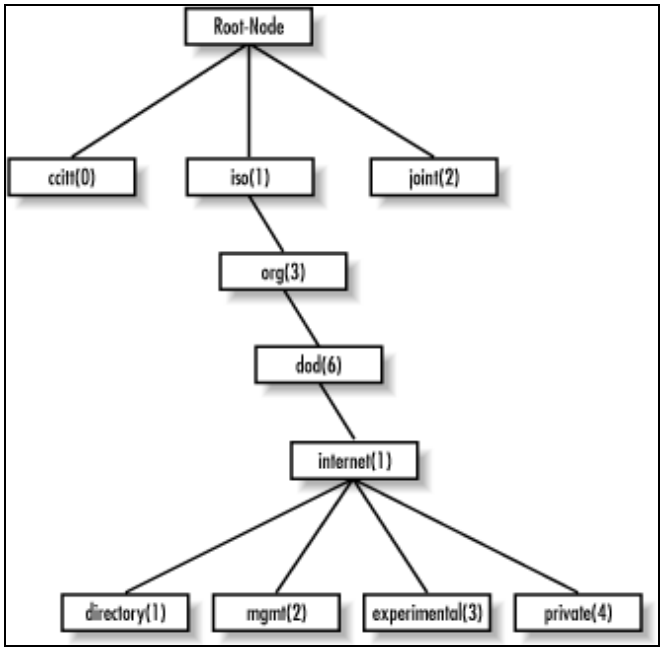

Figure 9 Structure of Management Information tree

The node at the top of the tree is known as the *root*. Nodes without children are called *leaf nodes*. Nodes with children are called a *subtrees.*

A standard set of Internet management objects are defined by the management branch (mgmt). The experimental branch is reserved for research purpose. The individual and organization use the private branch to define their objects.

Defining OIDs: It is a way of defining managed objects. The MIB value to retrieve the uptime of Cisco Router is 1.1.3.6.1.2.1.1.3.0.

3.5 SNMP Operations

The message format that managers and agents use to send and receive information is Protocol Data Unit. The PDU formats for each of the following SNMP operations are

- *get*
- *get-next*
- *get-bulk* (SNMPv3)
- *set*
- *get-response*
- *trap*
- *notification* (SNMPv3)
- *inform* (SNMPv3)

3.5.1 The get Operation

NMS initiates the get request and sends it to the agent. The agent processes the request and if it is successful in collecting the requested data, it sends a response back to the NMS. This process is illustrated in Figure 10

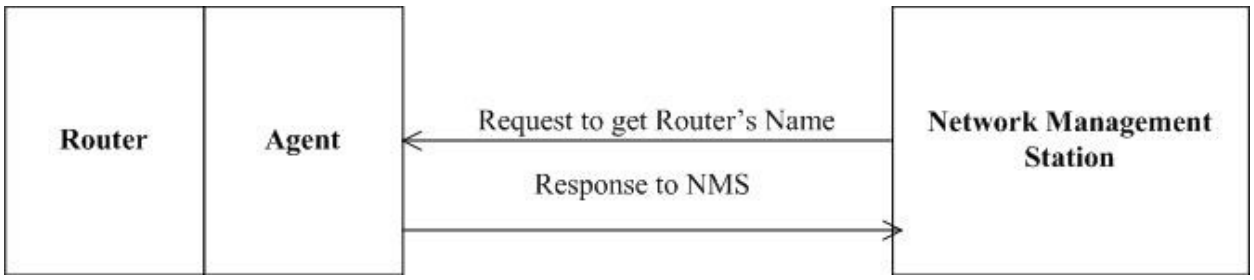

Figure 10 Get request command sequence

3.5.2 The get-next Operation

The get-next operation is used to retrieve a group of values from a MIB. Agents start at the root of the object tree and moves down till it finds the match for the OID.

The next get-next command is issued once the response for the previous command is received by the NMS and the process continues till the end of the MIB.

3.5.3 The get-bulk Operation

In order to retrieve a bulk data at once, get-bulk operation is used as it requests the agent to send as much response as possible. Two fields are set when issuing this command.

Non-repeaters and max-repetitions. The first N objects are retrieved using a get-next operation and the remaining are retrieved using M get-next operations.

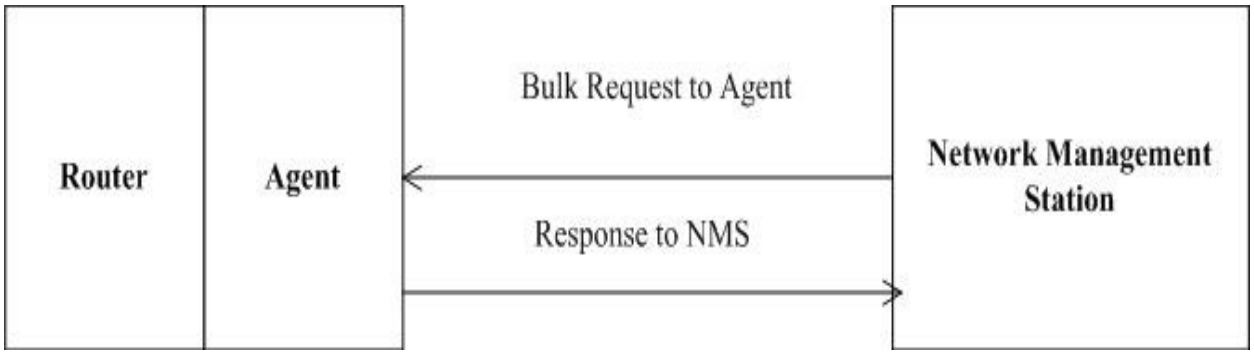

Figure 11 Get Bulk command sequence

3.5.4 The set Operation

To alter the value of a managed object, to create a new row, to alter read-write or write-only objects defined in the MIB can be accomplished using the set command.

3.5.5 get, get-next, get-bulk, and set Error Responses

Error response help determining if the agent processed the requests correctly.

3.5.6 SNMP Traps

A trap message is a way of informing the NMS that some mishap has occurred. Trap messages are originated from the agent and are sent to the IP address of the NMS. The NMS in return does not send any acknowledgement to the trap messages and hence the agent has no way of learning if the trap message has reached te destination or not.Since SNMP uses UDP there is no way to ascertain that the trap messages evn reach the NMS. Most of these messages are lost on their way to the NMS. If a NMS receives a trap, in order to facilitate it to intrepret the message, it carries a generic trap number. There are 7 such numbers counting from 0 to 6 as shown in the following table.

Table 1. SNMP Generic Traps

Generic Trap Name and Number	Definition
coldStart (0)	Agent reboot indicator. Counters and Gauges are reset to 0. The advantage of cold start trap is that it can be used ti identify if a new device is added to the network.
warmStart (1)	Agent reinitialize indicator.
linkDown (2)	Indicates that an interface in a device has gone down.
linkUp (3)	Indicates that an interface in a device has come up.
authenticationFailure (4)	Unauthorized access of community string
egpNeighborLoss (5)	Exterior Gateway Protocol neighbor down indicator.
enterpriseSpecific (6)	Enterprise specific trap indicator.

- **SNMP Notification:** The get and set have the same PDU format as that of NOTIFICATION-TYPE. Link down status is an example of notification type. The OID for this trap is ***1.3.6.1.6.3.1.1.5.3***
- **SNMP inform:** The "inform" mechanism which is similar to the "get" and "set" request is used for communication between two masters. This is useful when there is more than one NMS in the network. When a request is sent from one of the NMS in the network to the other, the other NMS sends an acknowledgement of receipt of the inform request.

3.6 Data Types

Table 2 MIB Data Types

Data Type	**Description**
Integer	32 bit number
Counter	32 bit number ranging from $0\text{-}2^{32}$. When maximum value is reached it starts over again.
Gauge	32 bit number ranging from $0\text{-}2^{32}$. A gauge can increase and decrease and can never exceed its maximum value.
Octet String	A string of zero or more octets is generally used to represent text strings.
Sequence	Defines lists that contain other data types
Sequence of	Defines a managed object that is built up of a sequence of ASN.1 types.
TimeTicks	32-bit number. Ranges from $0 - 2^{32} - 1$ (4,294,967,295). Time Ticks measures time in hundredths of a second

In policy based script MIBS, there are four components.

- Policy Manager
- Script Repository
- Agent Manager
- Target points

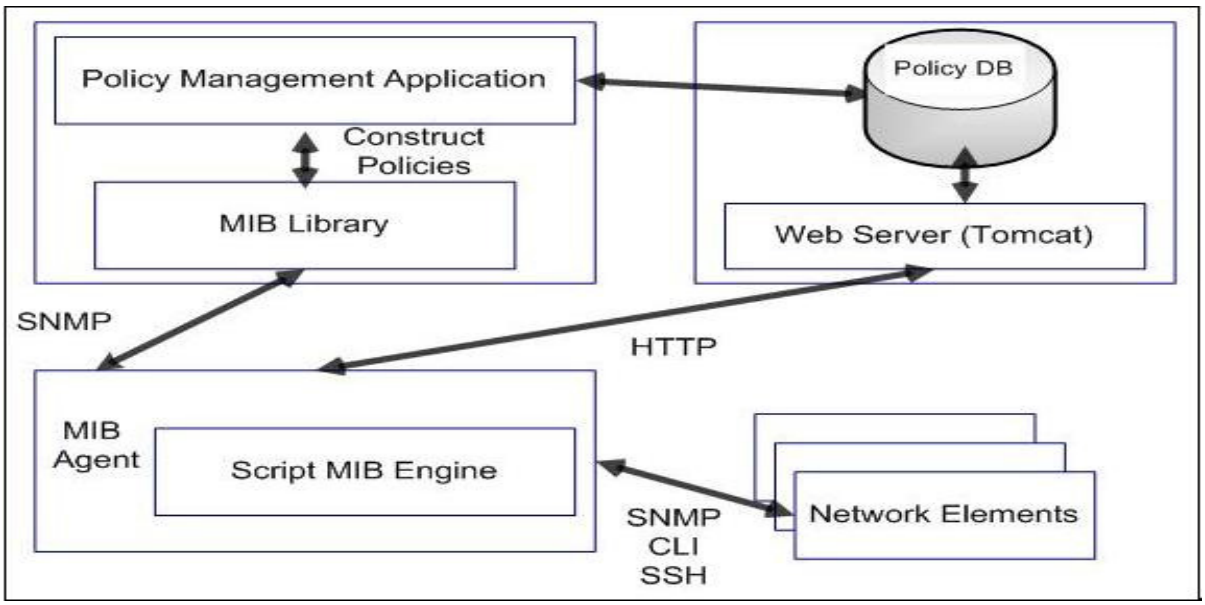

Figure 12 Policy based MIB script for Configuration Management

The policy manager controls the entire policy based management. It provides user interface which allows administrators to construct policies and store them in Policy DB. The policy management application keeps track of the roles of the network elements controlled by Script MIB. The script MIB is responsible for retrieving the policies from HTTP server. Target points are the network elements that need to be monitored/controlled. The communication between the agent manager and target points is via SNMP, SSH/CLI.

There are two ways in which script MIBS can be executed. One way is to run as a process and the other is to run as a class object.

This architecture also has disadvantages when used on distributed networks. Although the processing is done by agents as opposed to the servers, but still the concentration of policies on a single server does not eliminate the scalability issue. But with slight modification to this architecture we could resolve the scalability as well as the network congestion. The answer lies in use of mobile agents.

3.7 Mobile Agent

Mobile agent is an autonomous object that is suspended, serialized and transferred to remote node where it can execute.

Mobile agents are easy to program on remote nodes because of their migrating and transfer capabilities. Agents have the following attributes:

- Autonomous
- Adaptability
- Collaborative
- Reactive

Also the following benefits were seen in [26].

- Increase performance
- Lower overhead
- Asynchronous execution
- Fault tolerance
- Existing ideas on software mobility

- Approaches for network configuration management

Mobile agents play an important role in detection of faults.

3.7.1 Fault Detection & Management

Mobile agents help in identifying the faults in network and also reporting them to their respective managers.

Fault management is a process of developing a specialized network model. Mobile agents decentralize the process by checking for fault detection, ie; over-utilization of resources, error prone nodes. If the rules indicate violations of normal behavior of a network element, then the agent performs fault detection function.

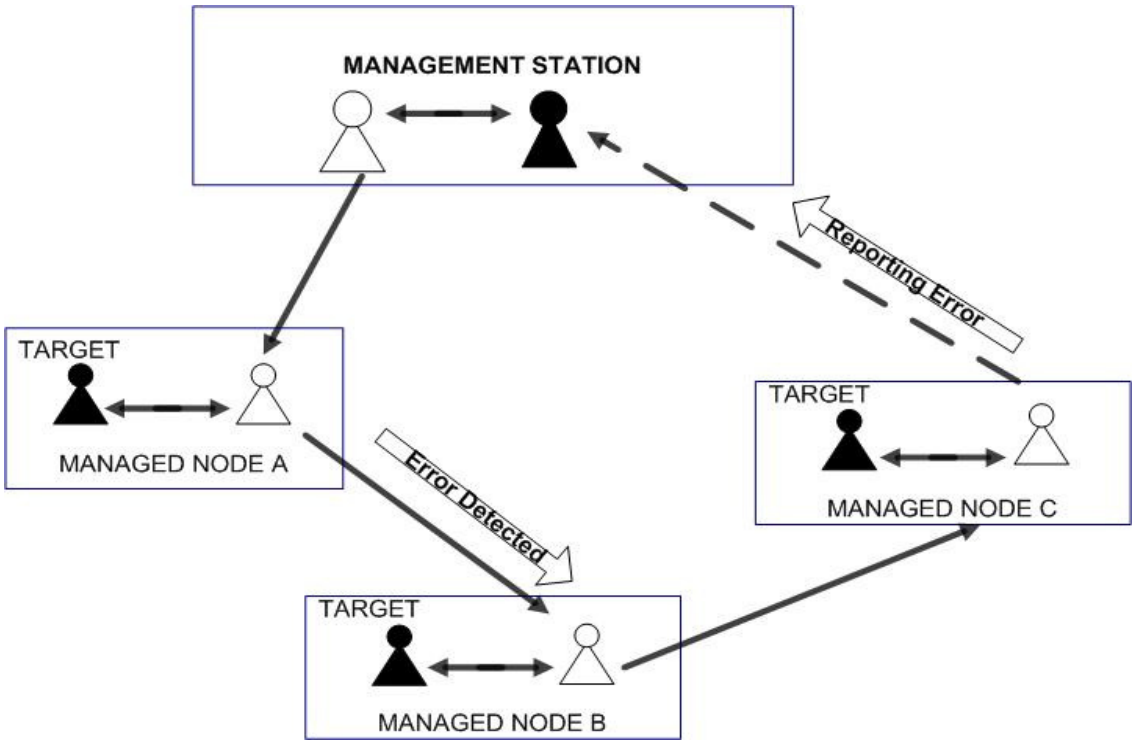

Figure 13 Error Detection and Correction by Mobile Agent

For example, when an interface goes down in node B, the router in node C sends a trap signal to the appropriate Network Manager. The Network Manager may take decision on its own

or may contact the higher level manager to get the appropriate MIB script. The mobile agents then carry the mobile code and execute on the remote end to resolve the fault. This kind of fault correction is known as code on demand (COD).

3.7.2 Mobile Agent Security

With mobile agents comes the question of security or how to differentiate if the agents are from trusted source. Keeping in mind to restrict access to sensitive information stored on network nodes, we need to have some kind of privileging level to access information.

Stationary agents

- Have access to system information like processes running currently, memory usage, open ports etc

Agent server accepts connections from other agents which creates a runtime environment. The runtime environment could be TCL, perl or java.

3.8 Platform Assessment

Agent with autonomous capability along with its intelligent adaptation to the environment is known as constrained mobility. Its termed so because agent intends to migrate where its environment is confined. Under constrained mobility whenever there is a change in requirement the model can program itself by replacing older agents. Constrained mobility is used in configuration management systems, fault tolerant etc. Agents like Grasshopper and agile can well be used for such purposes.

Aglets: It is written entirely in Java. It uses its own communication transfer protocol. It also has its own interface for communication.

Grasshopper: It's a commercial product with a free license. The entire software is implemented in Java. It supports a number of transport and communication protocols. It also has support for RMI as well as CORBA. Both the software is customizable to specific applications.

These agents have object based behavior but are dependent on management nodes. Agents like Grasshopper and agile lack the knowledge of requirement satisfiability.
But one thing to be noted in common is that the use of Java. Because of its full potential as an object-oriented language and also its security and portability features, Java clearly dominates over any other programming languages in implementation of mobile agents.

Stationary agents fulfil the requests from mobile agents which prevents the mobile agents from accessing sensitive information.

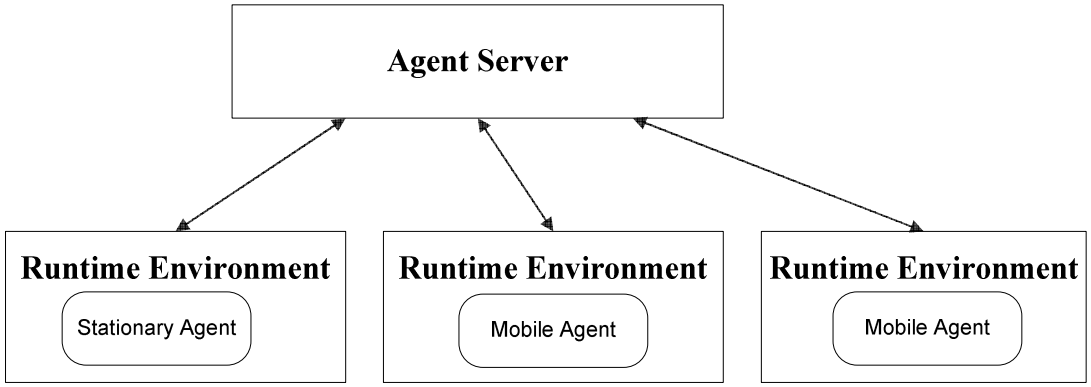

Figure 14 Agent Configuration

Management application requests the mobile agent to carry out the tasks.
In case of new requirement, the management application would transfer the configurations to new devices through the use of agents.

3.9 TEST BED

Figure 15 shows the overall functions of management station. In the following sections, we shall focus on configuration management covering Configuration Display, Import and Export of configuration, Validation of configuration.

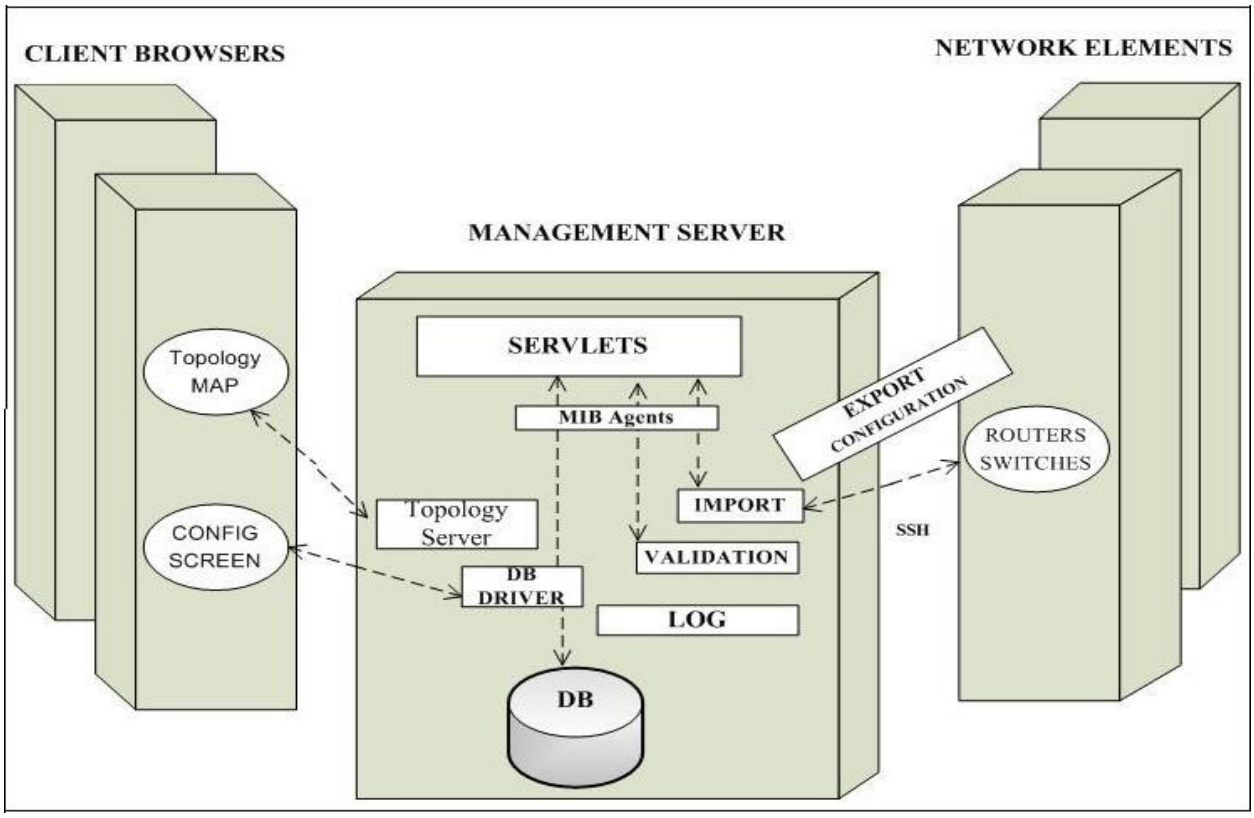

Figure 15 Interaction of NM Server with Client Browsers and Network Elements

The NM interacts with the network elements through SSH for import and export operations. The config display an GUI interface to all configuration information.Validation helps in checking the integrity of network configuration.

The GUI based explorer window simplifies the entire configuration process. And also security is enforced at every level. Secure shell is used between the user's interface GUI and

management server. The login key mechanism on the network elements is managed through RADIUS and TACACS.

The NM uses a special module for storing templates and the neighbour configuration files, so that it can be utilized for validation purpose. NM provisions the following:

- GUI based explorer windows for configuration display
- Checks syntax for configuration
- Integrity Checks
- Monitor/Audit configuration change
- Export/Import of configuration files

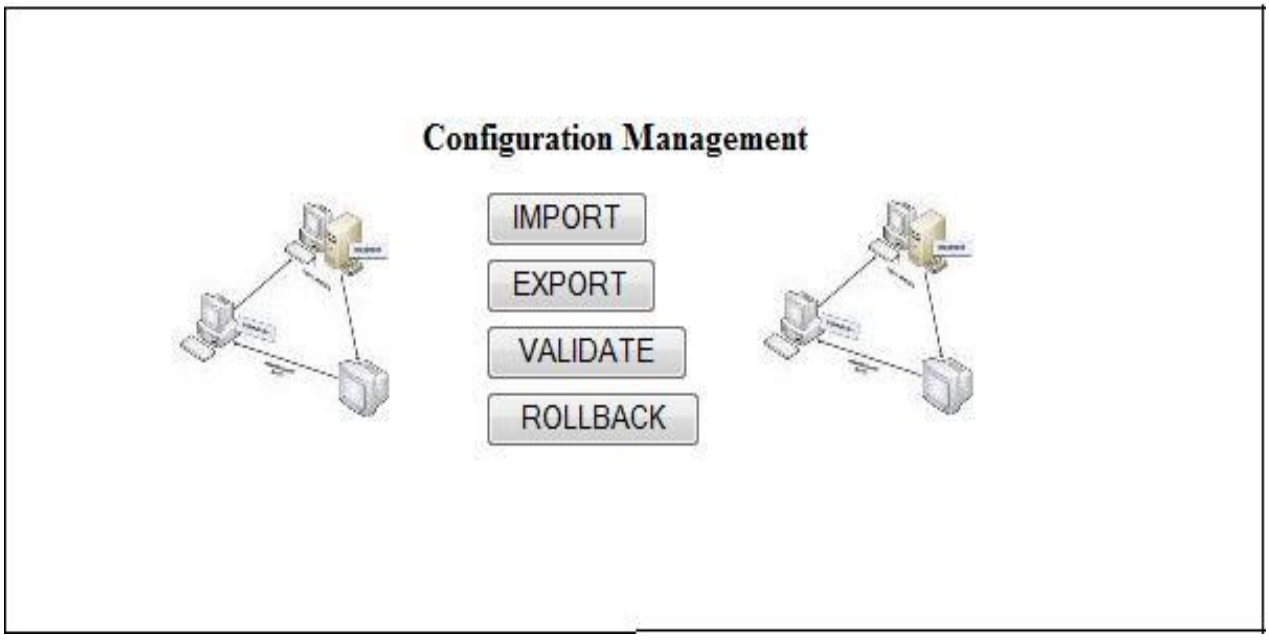

Figure 16 Configuration Management GUI

Menu items consist of the following icons.

- Import
- Export
- Validate
- Rollback

The following are the operations performed when the above buttons are clicked.

- Make a DB connection and retrieve the configuration file
- Validate the configuration and check for syntax errors
- Import configuration directly from the device or from TFTP server
- Export configuration directly to the device or to TFTP server
- Log messages to server
- Revert back to original configurations

3.9.1 IMPORT / EXPORT

The user shall be able to request from the configuration GUI, the configuration that shall be exported to or imported from corresponding routers. In case of an export request, the servlet will obtain the corresponding objects, attributes from the database and it will use the retrieved information to create a configuration file. The export command is then used to transfer the file from the NM server to the specified network element using SSH.

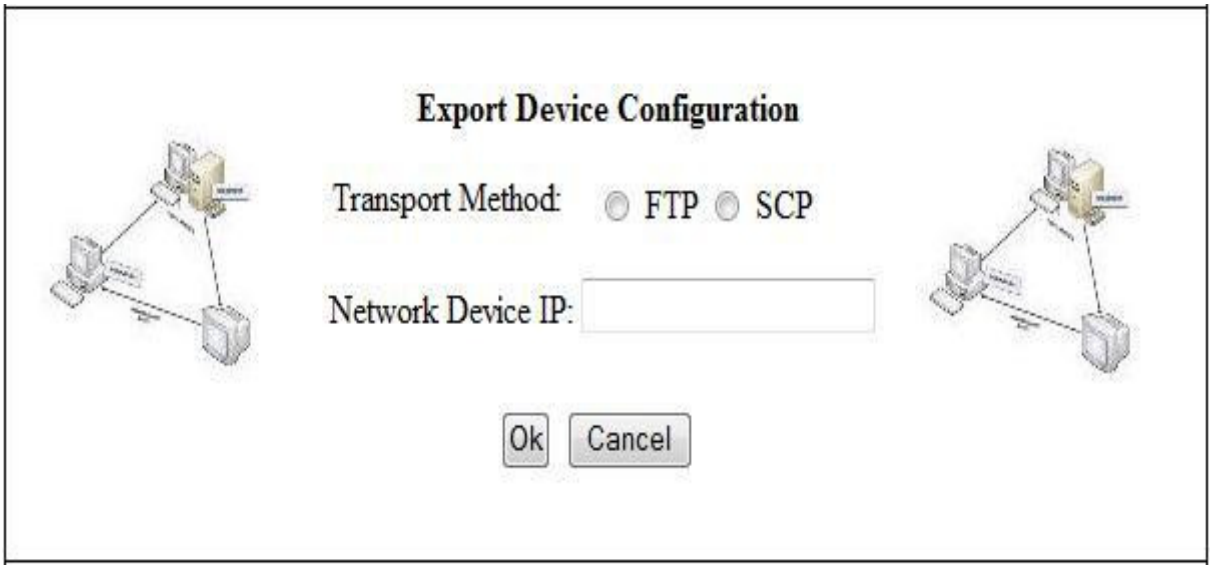

Figure 17 Import Device Configuration

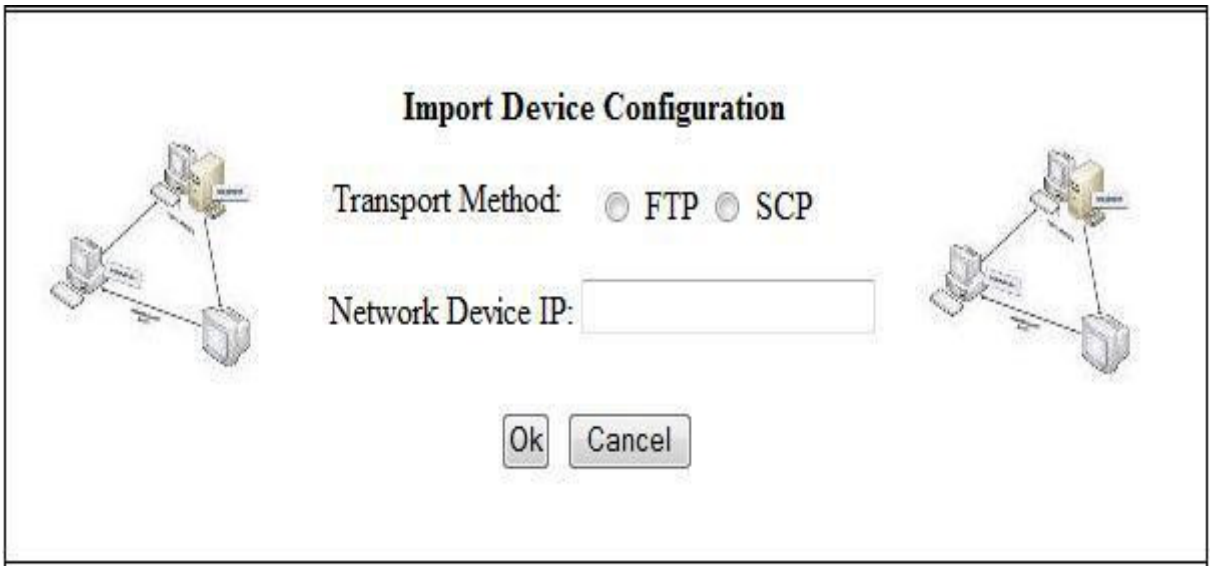

Figure 18 Import Device Configuration

The import function is very similar to export but in reverse order. The imported configuration file is written to the DB with new ID's. There are two transport methods provided because all devices do not support the same method. The transport method can FTP or SCP.

3.10 Modifying existing network using agents

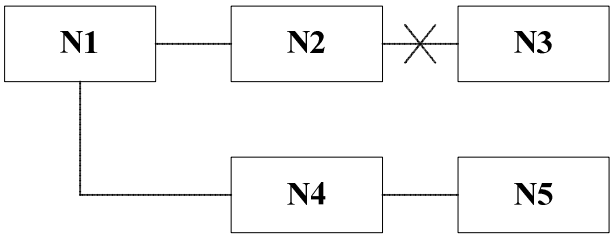

Figure 19 Mobile Agents modifying existing network

Lets say for example, the link between N2 and N3 is down, then N1 would sense it and would want to use the link N3. Agent would send appropriate commands to bring up the link between N4 and N5. Agents need not be run on all nodes. It is dependent on the network segment. In such cases the agents should have the capability to remotely login and be able to execute the commands. In either case the network fault is resolved and the network is back at normal operation.

Once the network is under normal operating condition, the agent running on N1 would send back the faults and commands that were used to rectify it to the management application. The management application does validation of the commands.

The validation function provides an broad integrity check detecting frequent yet hard to isolate pre-defined configuration issues by systematic diagnose. A high level script is developed for easy validation of configuration process. Many rules are defined in protocol parameters for integrity checks. Validation rules are defined for the following protocols

BGP, OSPF, RIP and IP addressing consistency.

For example,

BGP type of group mismatch.

OSPF timer value mismatch.

Example Script

Troubleshoot OSPF MTU Issues

```
 rule "Rule 3 : Unique Name for LSP " {
  message "The MTU check "+
   type "Rule 3 " ;
            protocol OSPF;
            severity medium;
find SystemlevelIP ( / / go through all routers
 set @MTU = [ ] ;
 set $addr = .RouterId;
  for all MTUInterfaces {
       if (.RouterInterfaceMTU in @MTU) {
            @MTU << .RouterInterfaceMTU;
        }
    }
}
```

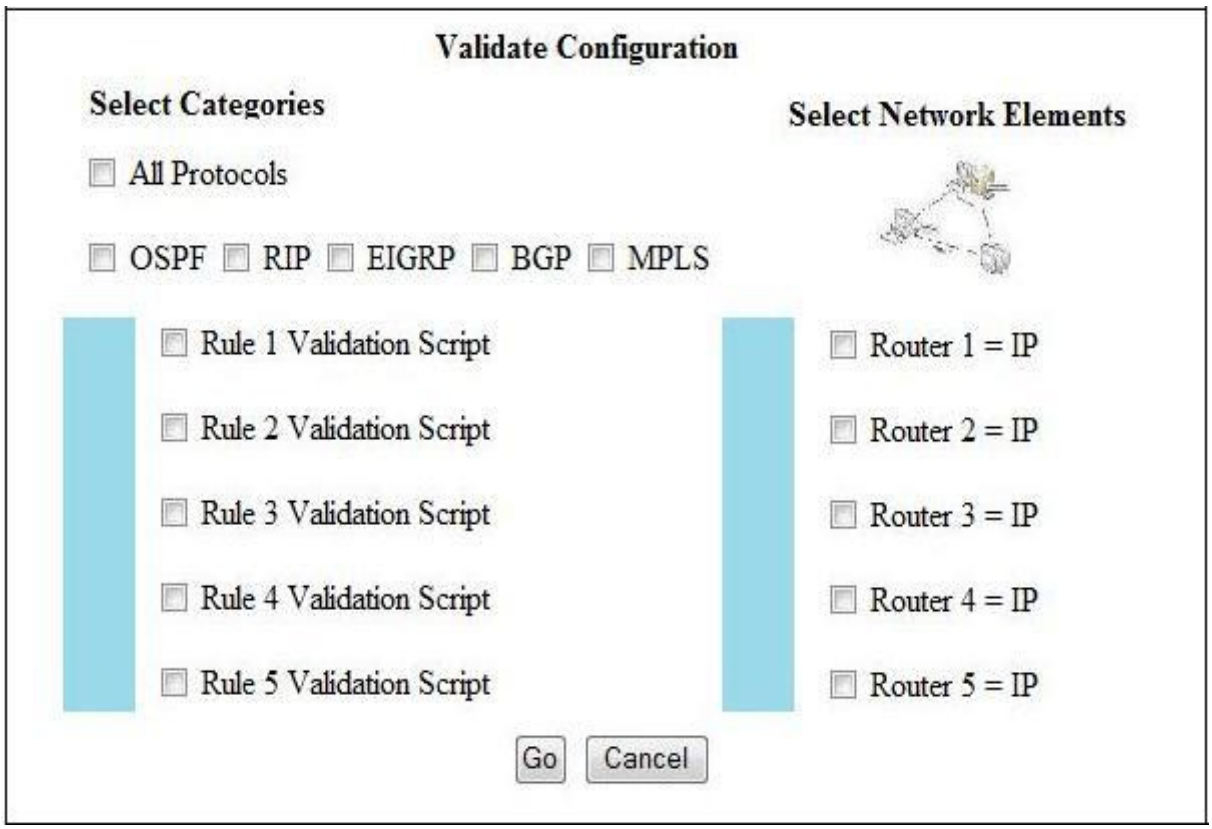

Figure 20 Validation of Configuration

There are many validation scripts that can be run on different routers. The validation scripts are run to check for consistency of configuration. For example, the Hello Interval values configured between neighbouring routers can be checked by running one of the scripts and selecting the suitable routing Protocol.

Thus the management application validates the faults and also has uptodate information on the network. This information would affect if there are any other new requirements that are to be fulfiled later.

CHAPTER 4

MATHEMATICAL MODEL

4.1 Intelligent System Modeler

The goal of Intelligent System Modeler described in [3] can be described mathematically using the following definitions. The goal of network configuration management is to divide the main requirement into smaller ones. This leads to multiple objectives that need to be solved to achieve the requirement. Also these individual objectives compete with each other for resources. With multiple objective scenarios we need to measure how optimal network configuration is addressed.

One approach as discussed in [40] is that each objective is given a value. These values are used to define a function that evaluates the success of respective specification. This approach is difficult as each objective has to be iterated through a long list.

So another approach as mentioned in [41] is to divide the main objective into sub-objectives based on the functional requirements. With this approach there are many interlinking of sub-objectives and cause difficulty in obtaining an optimal solution quickly.

In our approach, we slightly differ from the other two by adding constraints at each level of dependency encountered. This eliminates the number of iterations the algorithm has to go through.

N: set of nodes

C_i: current value of node i

F_i: Function to be applied to calculate c_i

R_i: Required value of node i

State of the network depends on variables of set V

$$\mathbf{V = \{ I \in N\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)}$$

Goal is defined by nodes with required values

$$\mathbf{G = \{ I \in N \} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)}$$

Depending on the choice of the variables in set of variables V, it can be found out if the goal is achieved or not.

Let constraints be a set of m variables $Y_1 \ldots Y_m$ each in a domain range of $R_1 \ldots R_m$. These domains represent the set of possible values.

S_r: Required value of node i

Take for a goal, G, to be achieved it is necessary that it satisfies all the constraints, that is $C_r = S_r$. This requirement puts restriction on set of variables Vg that decide the state of goal g.

$$\mathbf{V_g = J \in V \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)}$$

An assignment of values of V leads to all possible assignments c.

$$\mathbf{R_g = f \in \Gamma | C_j = f(j) \in V f \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)}$$

Domain of G is considered a subset of Γ where the function is a representation of Vg such that $C_r = S_r$

It is important to note that, not all goals produced by the Intelligent System Modeler would be achievable.

Decreasing the size of the domain reduces the complexity of the problem. Rg is further restricted by imposing (5).

$$f \in R_g \; V_f \in f(i) = h(i) \text{ …….. (5)}$$

Therefore, it is very important to have the constraints well defined, which in turn would have smaller number of domain range values. Once the optimal solution is found it is necessary to take appropriate actions.

Let A be the action required to achieve G. A is a variable with a value associated to it.

$$A = \{i \in V\} \text{…….. (6)}$$

Now we define a warning level associated for each action.

A_w Warning level of actions

$$V_w = i \in V \text{…….. (7)}$$

$$f_w(i) = f(i) \in V_w \text{…….. (8)}$$

By carefully evaluating the warning level associated the corresponding action can be taken. Hence the solution set $f_w(i)$ is obtained.

4.2 Mobile Code

Once the output from the Intelligent System Modeler is obtained it is necessary to realize the output in form of mobile code. Also to reduce the overall traffic and a way to accomplish decentralization, mobile code is used. This reduces the load on the management station.

In client-server architecture, client sends raw data to the server, and the server knows how to process the data and executes some services and produces some result which is delivered back to the client [27].

$$T_{traffic} = (2H + I + R)\, QN \text{…….. (9)}$$

In remote evaluation, node A knows how to perform the services but does not have the resources, so it sends the services to the node B which in turn executes the services and sends the

results back to A. We see that the overall traffic is much reduced when using mobile code under remote evaluation scheme.

Table 3 Parameters governing SNMP data retrieval

Parameter	Unit	Description
I	Bit	Size of SNMP instruction
H	Bit	Message header
R	Bit	Size of result of SNMP instruction
N_E	Elements	Number of Elements
Q	Instruction	Number of SNMP instructions to perform single device query

The overall traffic for mobile code including header is given by

$$\mathbf{T_{traffic} = (2H + I + RQ)N \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (10)}$$

"H" and "i" depend on the SNMP instructions; Q depends on the type of network and its functionality.

Thus by packing a set of SNMP instructions into mobile code we can avoid major overheads and network congestion.

4.3 Delay and Bandwidth Estimation

Most of the networks involve with movement of data from a source to destination. Configuration changes in the network would involve bursty traffic in the network. The traffic would be moving from the management station to the destination node. Under these assumptions, the mean delay can be modeled as in [35] and is given by

$$T = \sum \lambda_i / \Gamma T_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ (11)$$

This is a very generic equation.

In the above equation, it is assumed that the servicing time of each node is independent. But in our scenario, servicing time at each node is dependent on length of the data. So the

messages arrive at a Poisson rate and the servicing time is exponential, we can model the delay as a Markov model and is given by

$$\mathbf{T_i = 1 / (\mu C_i - \lambda_i)} \text{ ……………………….. } \mathbf{(12)}$$

T: Average Network delay (sec)

λ_i: Average traffic on channel i

T_i: Average delay in node i

μC_i: Channel capacity

After the mobile code has been developed, we derive mathematical equations to calculate the bandwidth utilization.

N_E - Number of elements to be monitored

N_o - Number of OIDs per element

Total Number of Elements = Ne*No……………………….. (13)

Assuming P to be the packet size, T_p to be polling interval and bandwidth BW_d is given by

P – Number of bytes in a Packet

24 x 3600 – Number of seconds in a day

T_p – Polling interval in sec

$BW_d = P \times (24 \times 3600) / T_p$ bytes/sec

BW_{util} = (Total number of elements) x P x (24 x 3600) / T_p bytes/sec

$$\mathbf{BW_{util} = (N_E * No) \times P \times (24 \times 3600) / T_p \text{ bytes / sec} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (14)}$$

$BW_{util\ ipv4}$ = (Total number of elements) x P_{ipv4} x (24 x 3600) / T_p bytes/sec

$$\mathbf{BW_{util\ ipv4} = (N_E * No) \times P_{ipv4} \times (24 \times 3600) / T_p \text{ bytes / sec} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (15)}$$

$BW_{util\ ipv6}$ = (Total number of elements) x P_{ipv6} x (24 x 3600) / T_p bytes/sec

$$\mathbf{BW_{util\ ipv6} = (N_E * No) \times P_{ipv6} \times (24 \times 3600) / T_p \text{ bytes / sec} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (16)}$$

In the above equation we have assumed P to be a single packet. But in distributed architecture packet size depends on the length of bulk transfer; let's say $P_{c,}$ and then bandwidth utilization is given by

$$\mathbf{BW_{util\ ipv4} = (N_E * No) \times P_{c\ ipv4} \times (24 \times 3600) / T_p \text{ bytes / sec} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (17)}$$

$$\mathbf{BW_{util\ ipv6} = (N_E * No) \times P_{c\ ipv6} \times (24 \times 3600) / T_p \text{ bytes / sec} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. (18)}$$

CHAPTER 5

SIMULATION AND RESULTS

5.1 PERFORMANCE MEASUREMENT

System performance is measured on client connection and response time from each subsystem. The testing environment is a 100Mbps Ethernet LAN. NM server is running on Linux box, with 1 GB RAM. Client machine is windows XP, Linux.

Table 4 Response Time for various Functions

Operations	Response Time (Sec)	
-	Ipv4	Ipv6
Client Connection	14	15
Import	7	10
Export	9	12
Validation	12	15

The table 4 shows the response time for operations like Import, Export and Validation using Ipv4 and Ipv6. The response time for various operations were obtained and plotted as shown in the graph.

5.2 GRAPHS

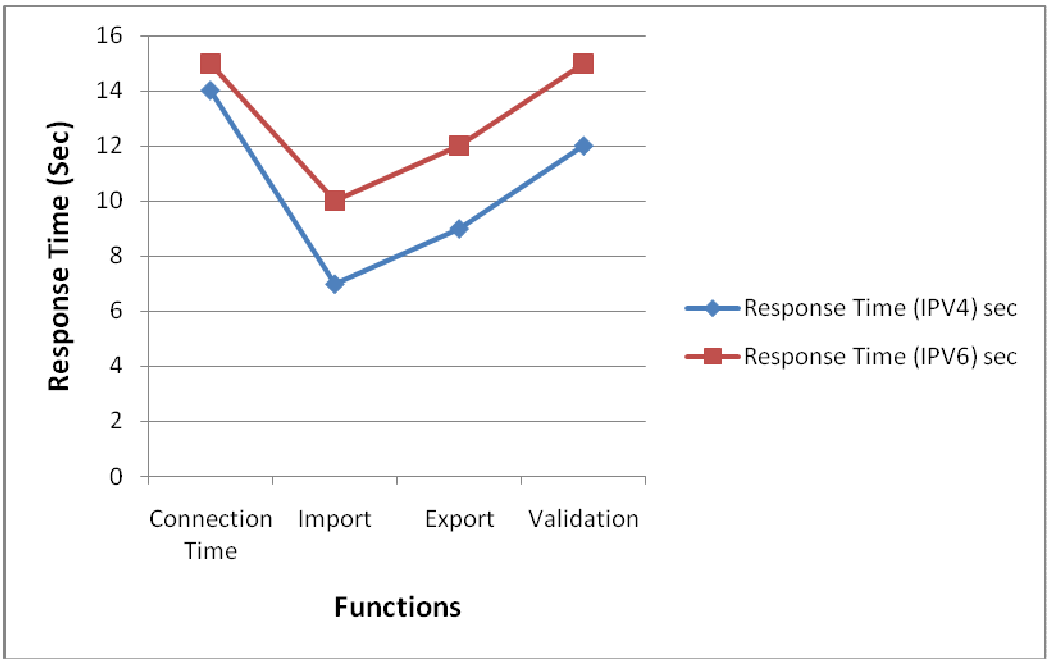

Figure 21 Response Time Vs Functions

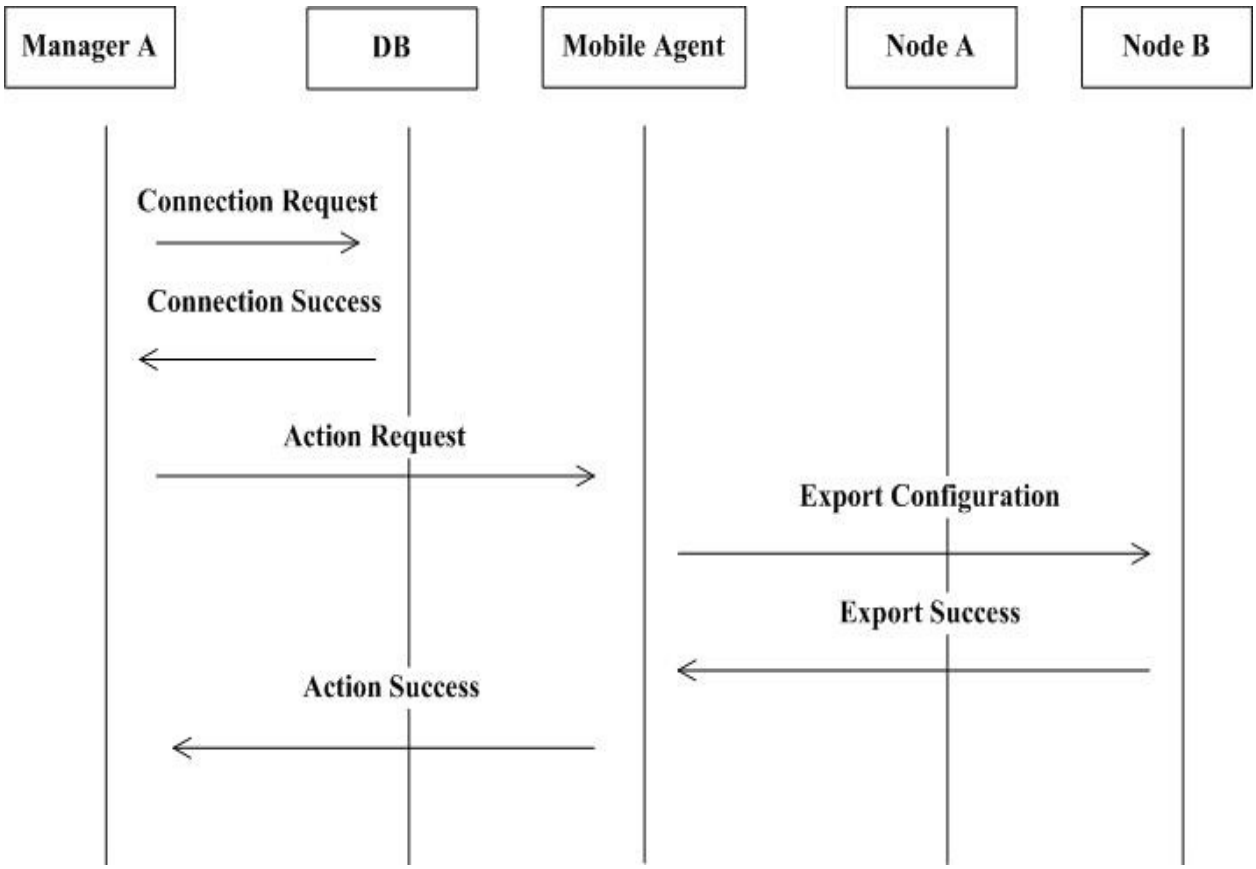

Figure 22 Connection flow between NMS and Nodes through Mobile Agent

The flow between manager A and nodes is shown in figure 22. The manager first establishes connection with DB and then sends a request to mobile agent. The mobile agent then exports the configuration to node B to make appropriate configuration changes. Once the configuration changes have been made, the node informs about the configuration changes made to mobile agent. Mobile agent then notifies NM (manager A) about the configuration changes.

The figure 23 shows the time taken for MIB scripts to be run on the agents with different number of elements with IPV4 and IPV6 packets. The MIB scripts were run to stabilize the network during network failure. It was seen that with increase in number of network elements the MIB scripts were efficient in terms of bandwidth.

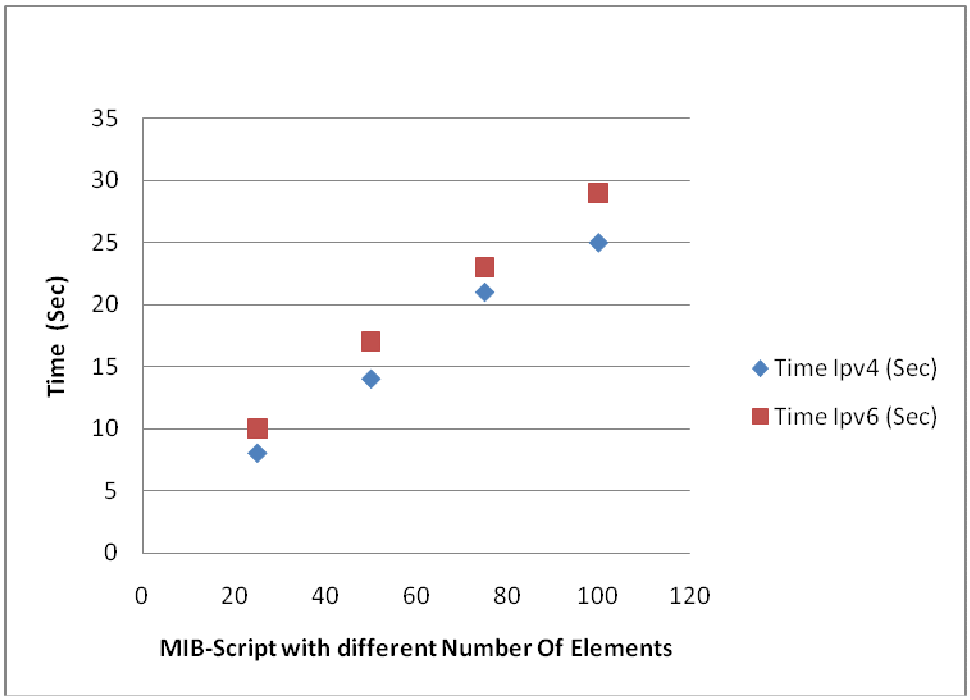

Figure 23 Number of Elements Vs Delay

The figure 24 shows the payload size for different number of elements. With increase in number of elements the traffic payload does not increase much because of the efficiency of the mobile code.

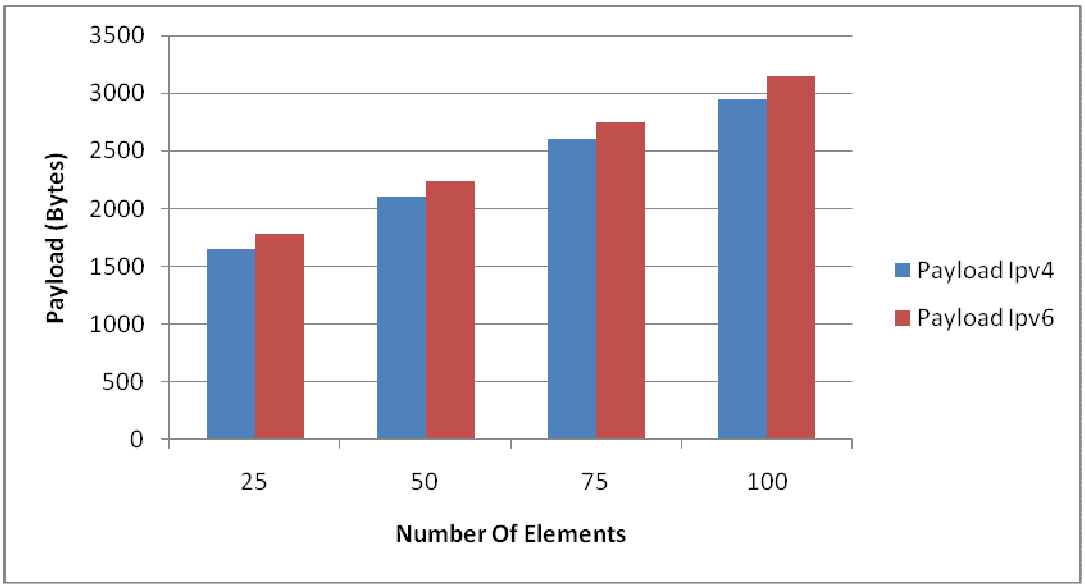

Figure 24 Number of Elements Vs Payload

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

With the proposed system, network manager can access and control the configuration of the entire network. Java Servlets and web based interface go hand-in-hand, that it can be smoothly integrated into the network elements for easy and efficient management. Java offers flexibility, high level of security and also reduced cost. Mobile code is a very promising solution for design and implementation of large scale applications. Script MIB's used for configuration management has proven to be feasible. Policies run as programs are flexible using standard programming languages. Simulation results have shown that managed station is capable of handling a large number of network elements up to few thousands.

6.2 FUTURE WORK

One of the issues with configuration management is, in what order the network elements should be configured. Development of newer algorithms in this direction would further enhance the Intelligent System Modeler. The current implementation has only a few levels of management stations. An extension is to increase the number of management stations and test it.

REFERENCES

LIST OF REFERENCES

[1] Haihong Gao, Jasti, A., Pendse, R. “Airplane Data Networks And Security Issues”, Digital Avionics Systems Conference, 2005.

[2] Joio Buptista Siu, Zhen Sheng Guo “Web-based Network Configuration Management System” Communication Technology Proceedings, 2000

[3] Md. Jakir Hossen, Abd Rahman Ramli, and Mohd. Khazani Abdullah “Web-based Network Device Management Using SNMP Servlet” Telecommunication Technology, 2003. NCTT 2003 Proceedings. 4th National Conference on Volume , Issue , 14-15 Jan.

[4] Hong Taek Ju, Mi Joung Choi and James W Hong, “An efficient and lightweight embedded Web Server for Web-based network element management”, lnternational Journal of Network Management, Vol. IO, Issue 5, September October 2000

[5] Wayne Jansen, Tom Karygiannis, “Mobile Agent Security”, http://www.genmagic.com/technology/techwhitepaper.html

[6] Glitho Roch H.; Olougouna Edgar ; Pierre Samuel, "Mobile agents and their use for information retrieval: A brief overview and an elaborate case study",Institute of Electrical and Electronics Engineers, New York, NY, 2002

[7] IBM Aglets, http://www.trl.ibm.com/aglets

[8] B. Brewington et al., “Mobile Agents in Distributed Information Retrieval,” Intelligent Information Agents, Matthias Klusch, Ed., Springer-Verlag, 1999

[9] G. Cabri, L. Leonardi, and F. Zambonelli, “Agents for Information Retrieval: Issues of Mobility and Coordination,” *J. Sys. Arch.*, 2000

[10] D.Kotz, R.Gray, "Mobile agents and Future of the internet", in ACM Operating Systems Review, August 1999.

[11] M.H. Guiagoussou, R.Boutaba, M.kadoch, "A java API for Advanced Faults Management", IEEE/IFIP International Symposium on Integrated Network Management

[12] Sun Microsystems, "The Java Lanaguage Specification", October 1995 java.sun.com/docs/books/jls.index.html

[13] Network Management, William Stallings, IEEE Computer Society Press, ISBN 0-8186-4142-8

[14] "A Web-Based Pro-active Fault and Performance Network Management Architecture", Andrea Silva Ramos, Anilton Salles Garcia, Rodolf da Silva Villaca and Rodrigo Bonfa Drago.

[15] "Network Management Tools", Mike Jude, Business Communications Review, May 2002

[16] JDBC: Java Database Connectivity, Bemard Van Haecke, IDG Books Worldwide, Inc, ISBN 0-7645-3 144- 1

[17] R. Fielding, "Hypertext Transfer Protocol-HTTP/I.O," RFC 1945, IETF HTTP WG, May 1996

[18] Servlet API Specification", http://java.sun.com/i2se/l.4/docs/~uide/

[19] Apache Software Foundations, www.apache.org/

[20] "Network Management Basics", http://www.cisco.com/en/US/docs/internetworking/technology/handbook/NM-Basics.html

[21] "Agent-based configuration management of distributed applications", Jurgen Berghoff, Oswald Drobnik, Anselm Lingnau and Christian Monch

[22] "Agent-Based Air Traffic Control in Airport Airspace", Vladimir Gorodetsky Oleg Karsaev Vladimir Kupin Vladimir Samoilov

[23] "Aircraft Configuration Management Using Constraint Satisfaction Techniques", D S Hill

[24] "Automated Network Management Using a Hybrid Multiagent System", Pere Vilà, Josep L. Marzo, Antonio Bueno

[25] "Simple IP Subnet VLAN Implementation", Chan Wai Kok', M. Salim Beg2

[26] "Network Performance Management Using Mobile Software Agents", Christos Bohoris

[27] "Understanding Code Mobility", Alfonso Fuggetta, Gian Pietro Picco, Giovanni Vigna

[28] "Web-based Configuration Management Architecture for Router Networks", Hosoon Ku, Jan Forslow and Joon-gil Park

[29] "An efficient and lightweight embedded Web server for Web-based network element management", Hong Taek Ju, Mi Joung Choi and James W Hong

[30] "Mobile Intelligent Agents for the Management of the Information Infrastructure", Stefan Covaci, Tianning Zhang, Ingo Busse

[31] Advent Network Management, Inc., Advent Java SNMP Package, http://www.adventnet.com/snmp_api.html

[32] Communication of the ACM, Intelligent Agent, Vol.37, No. 7,July 1994.

[33] T. Zhang, S. Covaci, "The Semantics of Network Management Information", Proceedings of the 1996 IEEE INFOCOM Conference, San Francisco, March 1996

[34] General Magic, Telescript Technology: Mobile Agents, http://www.genmagic.com/Telescript/Whitepapersh/wp4/whitepaper-4.html, 1996

[35] Jackson, J. R. 1957. Networks of waiting lines. Oper. Res. 5 518–521.

[36] J.Case, M.Fedor, M.Schoffstall, J.Davin, "A simple Network Management Protocol", RFC 1157.

[37] *Mi-lung Choi, Hyoun-Mi Choi, and James W. Hong, POSTECH Hong-Taek Ju* "XML-Based Configuration Management for IP Network Devices".

[38] Thanh Cheng, Brian Coan, Vikram Kaul, Kirthika Parmeswaran, William Stephens, "Building Autonomic Systems Via Configuration ".

[39] Morsy M. Cheikhrouhou, Pierre Conti, Karina Marcus, and Jacques Labetoulle, "A Software Agent Architecture for Network Management"

[40] Keeney R and Raiffa H, "Decisions with Multiple Objectives: Preference and Valiie", Wiley & sons.

[41] Kim I.S and Modarres. M 1987, Nuclcar Engineering Despatches,103

[42] Douglas Mauro, Kevin Schmidt, "Essential SNMP", O'Reilly Pub Date July 2001

[43] http://alloy.mit.edu/, MIT April 2009

[44] Network Configuration Management via Model Finding Sanjai Narain – Telcordia Technologies, Inc, LISA 2005