The Application of Intelligent Software Agents in Network Fault Diagnosis

Colm Davey BSc

A dissertation submitted for the degree of Master of Science in Computer Science

School of Business and Humanities

Institute of Technology
Sligo

Supervisor: Padraig Ryan

Submitted to the National Council for Educational Awards, August 2000

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other college.

Colm Davey

Colm Davey

August 2000

Abstract

The Application of Intelligent Software Agents in Network Fault Diagnosis
Author: Colm Davey

With the advent of existing and future networking, the task of network fault monitoring and in particular that of fault diagnosis is becoming quite complex.

Agents can be used to help in this domain. A software agent is a piece of software that is autonomous, goal-oriented and temporally continuous. Agents can be built on different architectures including reactive and deliberative. Agents can communicate with each other using languages such as FIPA's ACL or KQML.

Most network management applications currently use SNMP. SNMP is reliable but has a low degree of flexibility and is centralised. It was designed for less complex types of networks than those that currently exist. When comparing traditional network management applications with an agent-based applications, one needs to consider utilisation as well as fault diagnosis. Network utilisation is concerned with how a network is being used, and how much and what type of traffic is travelling through network segments.

When designing an agent-based network management system, it must be fast enough to react quickly to changes in the network. It should also minimise the amount of interaction that is needed with the network manager.

Acknowledgements

First and foremost thanks to my supervisor, Padraig Ryan, for support provided throughout the duration of the Masters. Thanks to Terry Young and Dermot Finan for providing me with teaching hours in the college. Thanks also to Paul Hannah of 3com for software provided which greatly helped in the testing of the agent application. Thanks are also due to Mike McGrath of LAN Communications for sponsorship and also to family and friends.

Contents

Figures and Tables

Chapter 1

Introduction

In today's information age, computers are prevalent in almost all aspects of society from entertainment to work to school and an infinite number of other personal interests. Over the last few years it has become more and more useful to have a computer attached to some form of network. For most home users this will be the Internet via the users Internet Service Provider (ISP). For people at work, this may be as simple as a basic office network consisting of just two PC's or it may be as complicated as a corporation with multiple wide area networks across continents with a different network for different job functions such as purchasing, sales and IT support. It is now even possible to "wear" a computer and form a network by simply touching someone else [36]. The emerging Wireless Access Protocol (WAP) technology also means that mobile phone users can now join the network revolution.

Although most users know they are connected to some form of network, they have no idea how the network works, only that it does. When it stops working it usually takes a skilled professional to solve the problem. This may be via a phone call or it may involve somebody needing to travel to the other side of the world. Network management usually requires full time, highly skilled professionals. As networks become larger and more complicated, the number of network technicians increases, as does the overall cost of managing the network. Even though network theory is well understood, it can be very time consuming to detect and solve network problems; serious network problems can cost companies huge amounts of money in lost time. Existing network management applications are usually based on protocols developed years ago (circa the 1970's) when networks were much smaller and less complicated. They were never designed to handle the vast array of networks that currently exist nor can they cope well with the new "global" aspect of networking technology, where networks are expected to be accessible 24 hours a day from anywhere in the world.

1.1 Aim of the Thesis

The aim of this thesis is to examine the use of software agents in helping diagnose and possibly solve network problems, with as little user interaction as possible. Software agents are still very much in their infancy but have shown much promise due to their autonomy, potential to communicate with other agents and ability to cater for distributed environments. It will be examined on a standard Ethernet LAN whether these agents have any promise for network management and fault diagnosis and also how they compare with existing network management applications.

- **Chapter 2** gives an introduction to what software agents are and also gives a number of accepted definitions. It also provides information on the current state of the art of software agents before discussing network management systems and their challenges.
- **Chapter 3** discusses how the agent management application was developed and the architecture that was used. It also probes inter agent communication and how to best store an agents knowledge.
- **Chapter 4** is concerned with how the agent software was set up on the test network, problems that were encountered and how they were solved. It

also discusses how the different agents interacted and decided among themselves how to best monitor the network.

- **Chapter 5** compares the agent software with SNMPc, a popular network management application that relies on the SNMP protocol.
- **Chapter 6** discusses the conclusions that were reached based on tests that were carried out. It also details possible future areas of research.

Chapter 2

Background Research

2.1 Software Agents Background

The concept of a software agent has been around for a long time in the Artificial Intelligence (AI) community. The concept of software agents began before the inception of Distributed Artificial Intelligence (DAI) in the mid to late 1970's. Kay points out that

"The idea of an agent originated with John McCarthy in the mid 1950's, and the term was coined by Oliver G. Selfridge a few years later... They had in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a 'Soft Robot' living and doing its business within the computer's world". [1]

2.1.1 A Software Agent Definition

Similar to the question "What is Intelligence", which has dogged the AI community for years, there is no completely accepted definition for what a software agent actually is. This has not been helped by the fact the when the term 'software agent' became more fashionable around 1994, many organisations jumped on the bandwagon by simply renaming existing software components as software agents. Franklin and Graesser [2] seem to have come up with a definition of software agents that is acceptable to most people. They mention a number of different agent definitions from a number of different people and organisations before coming up with their own definition that "An autonomous agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." Because, they point out, a thermostat satisfies the above definition they list some other properties that help further classify agents as follows:

Property	**Other Names**	**Meaning**
Reactive	Sensing and acting	Responds in a timely fashion to changes in the environment
Autonomous		Exercises control over its own actions
Goal-oriented	Pro-active	Does not simply act in response to the environment
Temporally Continuous		Is a continually running process
Communicative	Is socially able	Communicates with other agents
Learning	Adaptive	Changes its behaviour based on previous experience
Mobile		Able to transport itself
Flexible		Actions are not scripted
Character		Believable "personality" and emotional state

Table 2.1. Agent Classifications

They then point out that every agent should satisfy the first four properties. The other properties will determine the specific type of agent. They then specify a basic classification of agents as follows:

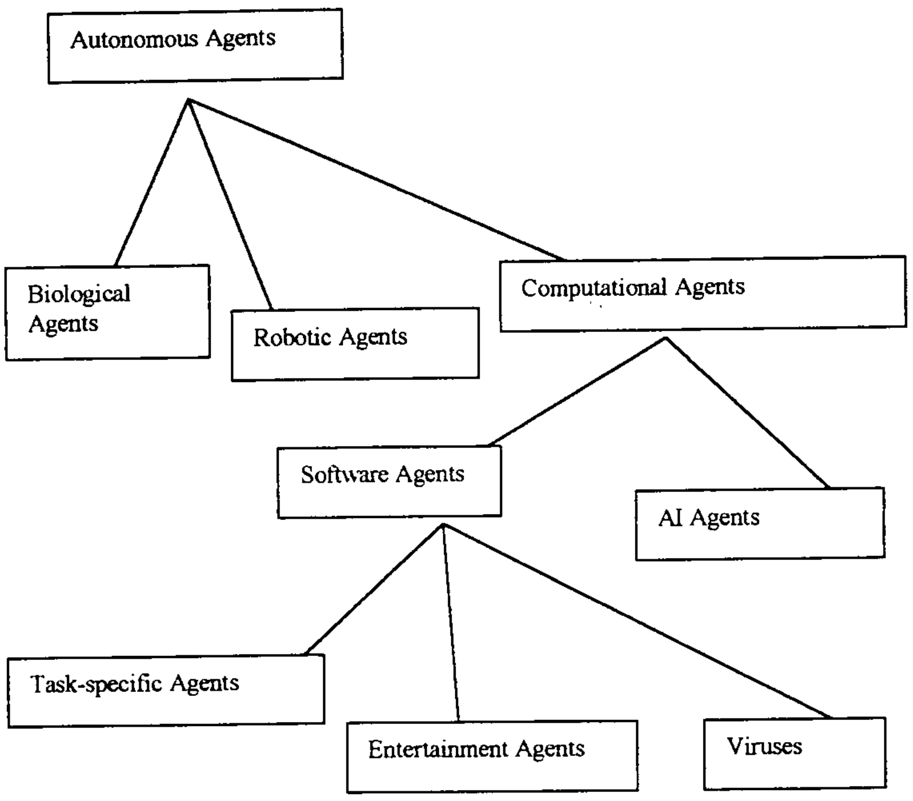


Fig. 2.1. Franklin and Graesser's Agent Taxonomy

Other authors have come up with similar definitions and characters that agents should have. Sycara et al [3] describe the following:

- **Taskable.** Agents can take direction from humans and other agents
- **Network-centric.** Agents should be distributed and self organising
- **Semi-autonomous.** Not under direct human control all the time
- **Persistent.** Capable of long periods of unattended operation

- **Trustworthy.** An agent should serve users' needs in a reliable way
- **Anticipatory.** An agent should anticipate user information needs through task, role and situational models
- **Active.** An agent should initiate problem solving activities
- **Collaborative.** An agent should collaborate with humans and with other agents
- **Able to deal with heterogeneity.** Both of other agents and information sources
- **Adaptive.** To changing user needs, and task environment

For Wooldridge [4], an agent should have the characteristics of autonomy, reactivity, pro-activeness and social ability.

Nwana [5] designed a topology of agents that classified agents according to:

- **Mobility**
- **Deliberative** or **Reactive**
- Exhibition of primary attributes which were **autonomy**, **cooperation** and **learning**
- Roles, as **information** or **internet**
- **Hybrid philosophies**, which combine two or more approaches in a single agent
- **Secondary attributes**, such as **benevolence**, **versatility** and **emotional qualities**

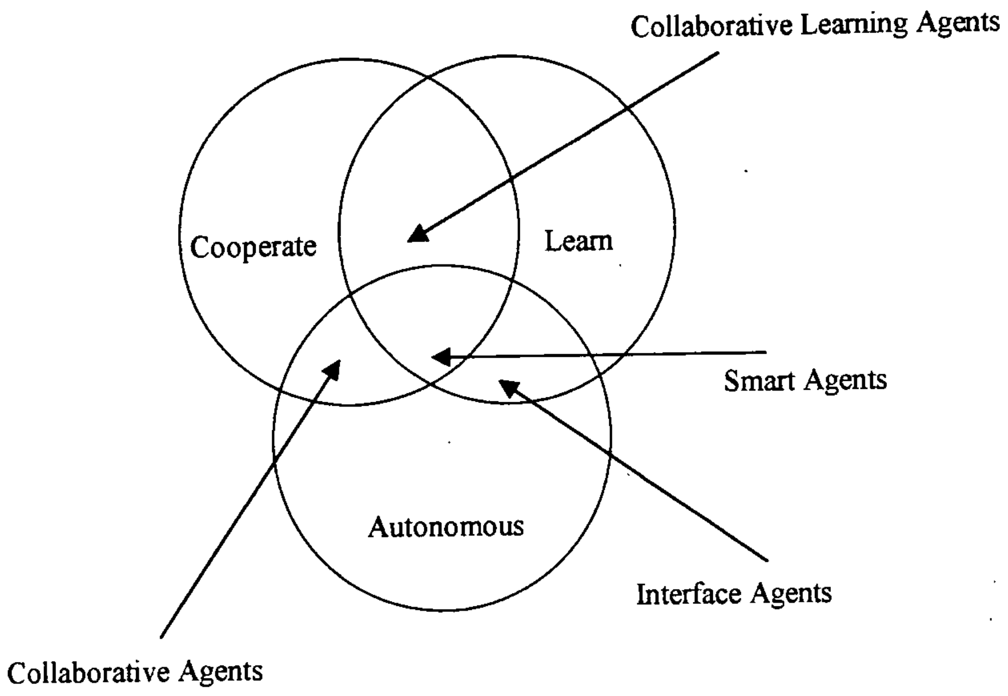


Fig. 2.2. Nwana's Primary Attribute Dimension

2.1.2 Agent Architectures

There are three main architectures used when building software agents.

A Deliberative Architecture [25] has a symbolic model of the agent's environment, usually represented in some limited subset of first-order predicate logic. There would also be a symbolic specification of the actions available to the agent and a planning algorithm which specifies how the agent can act in order to achieve a goal. Deliberative Architectures involve the solving of two important problems:

- The transduction problem. This involves translating the real world into an accurate symbolic description, in time for that description to be useful.

- The representation/reasoning problem. This involves symbolically representing information about complex real-world entities and processes.

Neither of these problems has been anywhere near solved even though there has been a large mount of research into them [24].

The agent's environment can be characterised as a set of *environment states* [6]

$$S = \{s_1, s_2, \dots\}$$

At any given instant, the environment is assumed to be in one of these states. The actions of an agent can be represented by the set

$$A = \{a_1, a_2, \dots\}$$

Thus the agent can be viewed as the function

$$\mathit{Action} : S^* \rightarrow A$$

which maps sequences of environment states to actions. This basically means that the action that an agent decides to perform depends on its history. The main problem with a deliberative architecture is that is that they do not scale to realistic scenarios i.e. the real world. Algorithms were based on *calculative rationality* [7]. Although calculative rationality will result in the best decision possible, that decision may not be made in time to be of any real use because of the possible size of the search space. This problem led many researchers to reactive architectures. Examples of deliberative agent systems are STRIPS [26] and HOMER [27].

A Reactive Architecture will enable an agent to decide what to do without reference to its history. They simply respond to changes in their environment and their behaviour can be represented by

$$\mathit{Action} : S \rightarrow A$$

Proponents of reactive architectures believe that intelligent behaviour is directly linked to the environment that an agent is located in. One of the most famous reactive architectures was Brooks' *subsumption architecture* [8]. Brooks pointed out that current deliberative architectures originated from research into mobile robots that used the *sense-model-plan-act* (SMPA) architecture. The SMPA framework involved simplifying a model of the real world and allowing robots to be designed in specific environments. There was an assumption that once robots could work effectively in the simplified world, a more dynamic environment could be tackled. This never happened. Brooks also pointed out what he believed were three key elements of organising intelligence:

- Most of what people do in their day-to-day lives does not involve problem solving, but routine activity in a relatively benign, but dynamic, world.
- An observer can legitimately talk about an agent's beliefs and goals, even though the agent need not manipulate symbolic data at run time.
- In order to really test intelligence it is important to build complete agents that operate in dynamic environments using real sensors.

Agre [28] also noted that most everyday activity is routine and he demonstrated his ideas with the PENGI system, which was a simulated computer game using a reactive architecture.

A Hybrid Architecture is basically a mix between a deliberative architecture and a purely reactive architecture. Using this architecture, there may be two or more subsystems (deliberative and reactive), with the reactive subsystem having precedence over the deliberative subsystem so as to be able to deal quickly with changes in the environment. A variation of the reactive architecture is the *layered* architecture [9]. Higher layers will deal with information in increasing levels of abstraction.

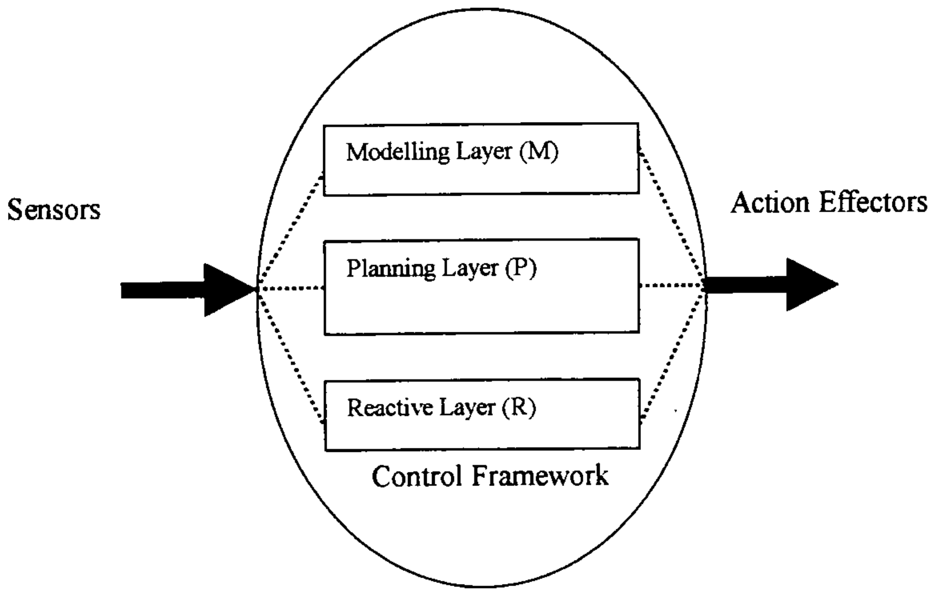


Fig. 2.3. Ferguson's TouringMachines Agent Control Architecture

In the above example, the reactive layer will generate potential courses of action in response to events that may happen too quickly for other layers to deal with. The planning layer will construct plans and select actions in order to achieve goals. The modelling layer will contain symbolic representations of the cognitive state of other entities in the agent's environment. These models can be used in order to resolve goal conflicts.

2.1.3 Current State of the Art of Software Agents

Although software agents have been promising much over the past few years there has not been a huge amount delivered. UK consultancy firm Ovum [10] predicted that the software agents industry would be worth US$3.5 billion by 2000.

Although there currently are no concrete figures available, judging by the lack of commercially available agent applications it is unlikely that the industry is worth anywhere near that figure. The OASIS system which Rao and Georgeff [11] had stated was one of the first to tackle a wide and complex real-world application, was being field tested at Sydney airport but has since been dropped due to legacy and political problems. The OASIS system used the BDI (Belief, Desire, Intention) model and was designed to help in air traffic control. It consisted of an agent for each incoming aeroplane; a sequencer agent that would determine which plane landed when, a wind modeller agent that dealt with wind issues, an overall coordinator agent and a trajectory checker agent. The primary objective of the system was to land all aircraft safely and in optimal sequence. Wooldridge [12] has pointed out that although the theoretical and experimental foundations of agent-based systems are relatively well understood, not much effort has been spent on practical agent systems development. There are a number of common errors that people and organisations seem to make when developing agent systems including:

- People believe that agents are a "silver bullet" (similar to AI) that will allow them to create applications with human-like understanding and acting. This is well beyond the state of the art. Nwana and Ndumu [13] argue that Maes's [37] seminal paper on agents for reducing work and information overload presumed that the intelligent tutoring problem had been solved but they point out that in retrospect this is not the case. They also argue that the nearest thing most people see that resembles a personal agent, Microsoft's ® Paper Clip, can be more annoying than useful.

- Agents get too heavily linked to AI. Too much time can be spent focusing on the agent framework part of the application and AI techniques to build that framework. Etzioni believes that using his "useful first" strategy would be better [14].
- Many researchers seem to spend time developing their own architecture or agent development language in the belief that that is the only way to develop their own application. Because not enough time is then spent developing the actual application, the project may flounder. Many applications may not even need a formal agent architecture and can be developed just as well in standard programming languages.

Jennings [15] points out that there is no data yet to show that agent-based computing is better than standard applications on a standard set of software metrics. It is generally accepted that the idea of an agent revolution has been replaced by an agent evolution. Although a lot of work has been carried out on agent communication [16], [17] there a choice between Knowledge Query and Manipulation Language (KQML) and the Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (ACL) and neither is perfect. For the moment, KQML exists in more applications but different KQML implementations cannot interoperate, which was one of the dreams of distributed software agent developers. Some researchers have discussed agent honesty and negotiation between agents but this becomes quickly irrelevant in many cases if agents belonging to different organisations cannot communicate properly. However, it should be noted that the foundations of Agent Communication Languages (ACL's) are quite strong.

2.2 Network Management Systems

2.2.1 Ethernet Networks

Ethernet is the most popular LAN technology and was developed in the 1970's by Xerox Palo Alto Research Centre (PARC). It is based on Carrier Sense, Multiple Accesses with Collision Detect (CMSA/CD) and it supports transmission rates of 10Mbs, 100Mbs or Gigabit. In theory, any frame sent by any station on an Ethernet network will reach and can be read by all other stations but this is not the case if the LAN is split into segments. Segmentation is the process of separating certain portions of network traffic, either for performance, security, or reliability reasons. A segment will only receive broadcast frames and frames destined for a particular device on that segment. If segmentation is done via a hub, all devices on that hub will be able to see all traffic that passes through the segment. If segmentation is done via a switch, the device attached to the switch will only be able to see traffic destined for itself and broadcast traffic. A hub is a device which will retransmit frames received on all outgoing ports, regardless of the destination of the frame. A switch will check all incoming frames and will only transmit the frame via the port on which the device resides. When a network adapter has a frame to transmit it checks if the line is idle (Carrier Sense). If it is then the adapter sends out the frame immediately. If the line is busy it waits until the line is idle and then transmits the frame. In a standard Ethernet network, once an adapter has sent a frame it must wait at least 51 μs before it can send out another frame. This gives other adapters the chance to send frames. Ethernet is a 1-persistant protocol due to the fact that an adapter with a frame to send transmits with probability 1 whenever a busy line goes idle. Due to the fact that no one adapter has control of the network, two or more adapters may send a frame at exactly the same time. This causes a collision of the frames that usually results in Jabber. Jabber is where two or more frames collide resulting in a larger frame that contains no meaningful data. Because Ethernet supports Collision Detect, the adapters will be able to detect that their frames are colliding. Once an adapter detects a collision and stops transmitting, it waits a certain amount of time before trying to transmit again. It will continue to do this, each time doubling the amount of time it waits, until the frame has been successfully transmitted. This is called *exponential backoff*.

A standard IEEE 802.3 frame consists of the following:

- Preamble, 7 Bytes
- Start of Frame Delimiter, 1 Byte
- Destination Address, 6 Bytes
- Source Address, 6 Bytes
- Length of Data, 2 Bytes
- Data, Between 46 and 1500 Bytes
- Frame Check Sequence, 4 Bytes

An Ethernet II frame consists of the following:

- Preamble, 7 Bytes
- Start of Frame Delimiter, 1 Byte
- Destination Address, 6 Bytes
- Source Address, 6 Bytes
- Type, 2 Bytes

- Data, Between 46 and 1500 Bytes
- Frame Check Sequence, 4 Bytes

After the Network Interface Card's (NIC) drivers have removed the preamble, start of frame delimiter and the frame check sequence field, it can be determined that an Ethernet frame must be of a size between 60 and 1514 bytes in length.

Although the length field usually refers to the length of the data field, if this value exceeds 05DC Hex (1500 Decimal) it indicates that the field is a type field and the frame is an Ethernet II frame. This can then be used to indicate the higher-level protocol contained in the data field. The following are common examples of Ethernet type field assignments:

0800	IP
0806	Address Resolution Protocol
8137 – 8138	Netware IPX/SPX
814C	SNMP

The source and destination address refer to the Media Access Control (MAC) address as opposed to an IP address. Each network card comes with its own MAC address as a hexadecimal address in the format **XX:XX:XX:XX:XX:XX**

The first three bytes refer to the manufacturer of the card and the last three bytes refer to the Network card ID. No two network cards can have the same ID. Below are some network card manufacturers along with their unique ID's:

00-10-5A	3COM CORPORATION
00-10-FE	DIGITAL EQUIPMENT CORP.
00-10-7B	CISCO SYSTEMS, INC.
00-00-09	XEROX CORPORATION
00-00-1B	NOVELL INC.
00-C0-4F	DELL COMPUTER CORPORATION

Link control information is held using the IEEE 802.2 standard in the data field of the IEEE 802.3 frame. The control information is carried in a Logical Link Control (LLC) Protocol Data Unit (PDU) as follows:

- Destination Services Access Point (DSAP)
- Source Service Access Point (SSAP)
- Control
- Information

The DSAP and SSAP fields are enclosed in the first two bytes of the data field and are normally used to indicate Service Access Points (SAP). SAP's act much like a mailbox and provide a mechanism for exchanging information between the Logical Link Control (LLC) layer and the MAC and network layers. The DSAP and SSAP fields should always match and examples are as follows:

00	Null SAP
06	IP
AA	SNAP
E0	Novell Netware
F0	IBM NetBIOS

The control field contains information concerning the type and class of service being used for transporting LLC data.

2.2.2 SNMP

The most widely used network management approach is based on the Simple Network Management Protocol (SNMP) [18], which comes from the Internet Engineering Task Force (IEFT). Another less popular approach is the one suggested by the International Standards Organisation (ISO), which is based on the Common Management Information Protocol (CMIP) [19]. Both of these approaches involve a low degree of flexibility and are centralised. They can generate a lot of congestion around the management station which has to deal with almost the entire computational burden and it has been proven that they lack scalability [20]. Both approaches are quite similar in that a management station (usually operated by the network manager) will interact with SNMP agents running on network nodes. The SNMP software can operate on workstations but will usually operate on bridges, routers and switches. It allows the management station to access information about the device they reside on and will store its information in a Management Information Base (MIB), or in the case of the ISO approach, a Management Information Tree (MIT). Both SNMP and CMIP adapt a Client Server approach, with the management station acting as the client. The client interacts with the SNMP software through a management protocol that specifies the packet format for a set of basic operations. The management station gets small pieces of information from the agents as required and all of the processing is done at the management station. A standard MIB-I consists of the following groups [21]

Group	**Description**
System	Provides vendor identification and the time since the management portion of the system was last reinitialised.
Interfaces	Provides single or multiple network interfaces that can be Local or remote
Address Translation Table	Provides a translation between the network address and the physical address
Internet Control Message Protocol (ICMP)	Provides a count of ICMP messages and errors
Transmission Control Protocol (TCP)	Provides information concerning TCP connections, transmissions, and retransmissions
User Datagram Protocol (UDP)	Provides a count of UDP datagrams transmitted, received and undelivered
Exterior Gateway Protocol (EGP)	Provides a count of interrouter communications, such as EGP locally generated messages and information from EGP Neighbours
Internet Protocol (IP)	Provides count of IP packets and errors

Table 2.2. Standard MIB-I groups

SNMP has a core set of five Protocol Data Units (PDU's). These are:

- GetRequest: This retrieves a single value from an agent's MIB.
- GetNextRequest: This is used to walk through the agent's MIB table.
- GetResponse: This is what an agent responds to with either of the above two commands.
- SetRequest: This allows a manager to alter an agent's MIB.
- Trap: This command allows an agent to inform a manager about something which requires their attention.

SNMPv1 had some limitations, the main one being the fact that it lacked adequate security. It is possible to alter MIB's and also interfere with network devices. Routing tables can be altered and this limited the use of SNMPv1 in many cases to non-critical networks. SNMPv1 also lacked proper message authenticity and encryption. SNMPv2 was an improvement over SNMP and it included added security features to avoid unauthorised use of MIB's and added encryption to prevent unauthorised changes to devices such as the routing tables of switches. SNMPv3 added security levels which could be adjusted depending on the user. It also ensured robust message integrity, authentication and encryption. SNMP is quite easy to implement because its design is so simple. It is also in very wide use today with almost all major network vendors supporting it in their devices. However, it was never designed to deal with the complexity of today's networks and the information it provides is very basic. Also, even though it provides a huge amount of information in the MIB's, for most network managers only a small amount of this information is of any use. Therefore network professionals often spend a large amount of time searching for and filtering information.

2.2.3 RMON

Remote Monitoring (RMON) is a logical evolution of SNMP and can allow the management of network segments anywhere in the world. Its operations are based on software or firmware operating in managed devices or managed stand-alone hardware probes. Each managed device responds to network management station requests, which are transported via the SNMP protocol.

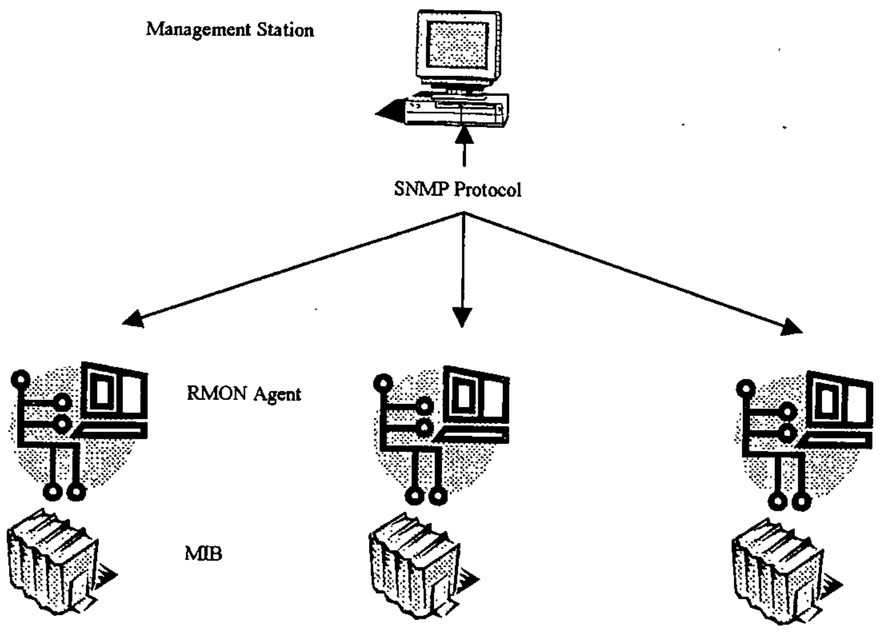


Fig. 2.4. RMON

RMON probes and devices will send statistics and alarms to a network management station upon request, or may generate a trap command. RMON can be considered a more traffic-oriented approach [22] than SNMP and CMIP because it directly inspects packets, instead of the status of individual devices. It therefore makes the network management approach mode decentralised. An RMON MIB consists of the following groups:

Group	**Description**
Statistics	Contains statistics measured by the RMON probe for each monitored interface
History	Records statistical samples from a network for a selected time interval

Alarm	Retrieves statistical samples on a periodic basis from variables stored in a managed device, and compares their values to predefined thresholds. If the threshold has been exceeded, an alarm is generated.
Host	Contains statistics associated with each host discovered on a network.
HostTopN	A group used to prepare reports that describe the hosts that had the largest traffic or error counts over an interval of time.
Matrix	Stores statistics of traffic and errors between sets of two addresses.
Filter	Permits packets to be matched based on a filter equation.
Packet Capture	Permits packets to be captured after they flow through a channel.
Event	Controls the generation and notification of events from the managed device.

Table 2.3. RMON MIB

2.2.4 Challenges for Network Management Systems

Davidson, Hardwicke and Cox [23] have identified three key challenges for current network management systems:

- Lengthy development life cycles. Many systems comprise of a small number of large software components that have long and expensive life cycles associated with them. The level of investment in each component results in a resistance to discarding it.

- Too much human involvement. Network management still requires skilled and expensive human operators who may not be able to cope with increasing complexity and dynamism of networks and services
- Architectures that do not scale. Centralised systems do not scale well.

Conclusion

It is obvious from the above that although agents are a relative young technology they offer many advantages that should be beneficial to the area of network management. Although there is no completely accepted definition, this should not present a problem as the fundamentals of agent technology and architecture are well understood. It is also clear that current network management approaches offer problems that the very nature of agents may be able to solve. These include the inherent distributive nature of software agents and also the fact that with proper temporal continuity and goal-oriented behaviour, they should reduce the need for expensive human interaction.

Chapter 3

Developing the Application

In order to develop an agent application a number of factors need to be taken into account. Although the application was developed for a relatively small network, it would be advantageous for the agent architecture to be able to deal with larger networks if required. The architecture would need to be flexible enough to handle immediate problems quickly without compromising overall system goals. A method of storing the agent's knowledge needs to be considered, as well as agent communication.

3.1 Deciding on an Agent Architecture

3.1.1 The network used

The network that the application was developed and tested on is a fast Ethernet 100 Mb/s computer network. There are approximately 400 PC's on the network with between 1 and 32 PC's on each segment.

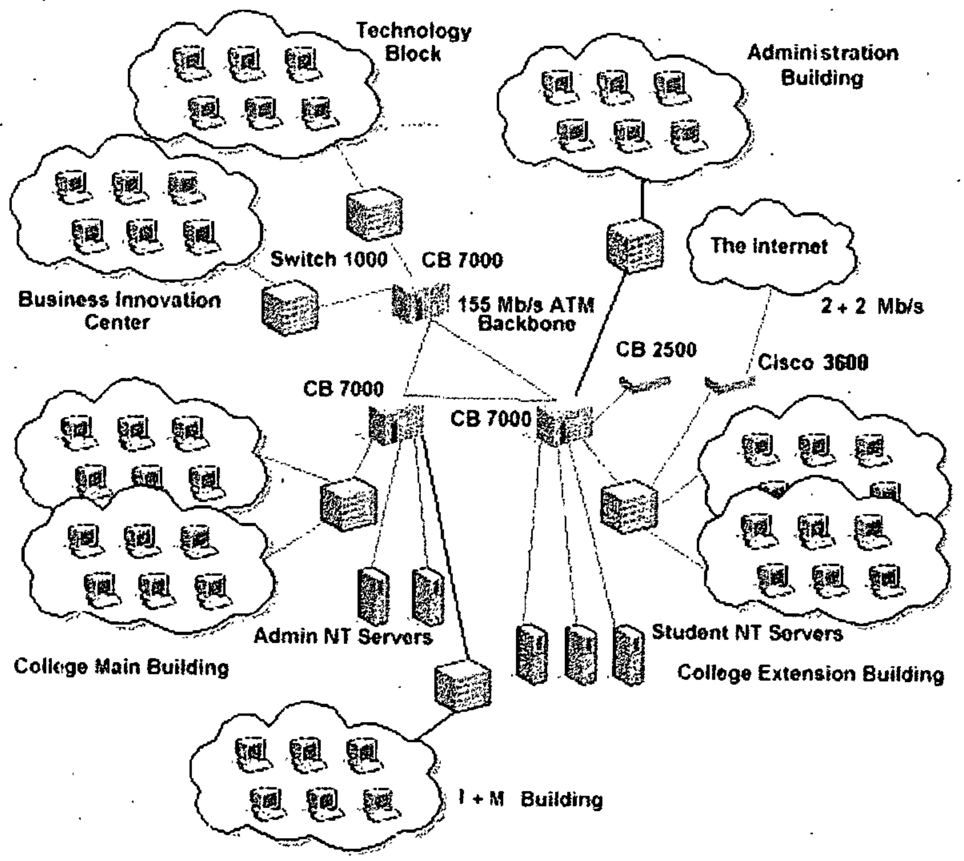


Fig. 3.1: Test network used

The network comprises of a 155 Mb/s ATM backbone and various switching equipment, all of which supports SNMP. The computer labs consist of either 16 or 32 PC's in a segment. A link from a switch connects to a hub with either 16 or 32 ports and each of these ports connects to a PC. Each segment has access to a bandwidth of 10Mb so obviously the smaller the number of devices on a segment, the more bandwidth a device will have. Although the switching devices support SNMP, they do not have the ability to have third party software installed on them, therefore it is not possible to place software agents on the switches and manage the network in that way. This is the case with most current switching hardware. The hubs in use on the network do not support SNMP so there is no way to directly talk to them. There are also segments consisting of between 1 and 5 devices. These are linked to a hub or directly to a switch and are used by administration staff. These segments are not regarded as important as computer lab segments when determining network utilisation. It is then obvious that the software agents need to be placed on the individual PC's. This led to a question of whether or not to use mobile agents.

3.1.2 Mobile Agents

A mobile software agent is basically the same as a standard software agent except for the fact that it can travel from one hardware device to another when required. These devices would usually be PC's and some switching devices. When a mobile agent is executing on a given network node it is able to migrate autonomously to a different node and then continue executing seamlessly as shown below [22].

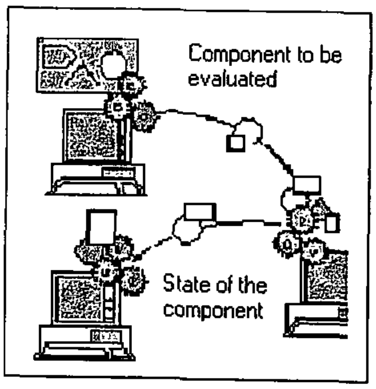


Fig. 3.2. Mobile Agent

However, there are very few switches that currently support mobile agents, which mean the agents would be required to run on a PC. Each PC that needs to be accessible by a mobile agent would require software running in memory in order for

the agent to be able to install itself and run on that PC. A specialised Mobile Code Language (MCL) would also be beneficial such as Telescript [29] or Agent Tcl [30] as these would make it easier for the agent and its state to move from one PC to another. At the time of writing there are no commercial applications using mobile agents and Nwana & Ndumu [13] have gone as far as to say that
"mobile agents are a clearly still a solution with no clear problem".
The mobile agents offered no advantages over the architecture chosen, especially because of the limited size of the test network, but had the disadvantages of longer development time and less evidence of reliability. It was for these reasons that they were not used.

3.1.3 The Agent Architecture Used

Because of the nature of Ethernet networks, a number of assumptions could be made when deciding on the agent architecture:

- The agents would need to be able to react quickly to changes or problems in the network. This type of problem lends itself to a reactive architecture.
- Most problems that can occur with Ethernet networks are already known. This means that the ability for a software agent to learn would not be essential. A given set of rules could specify how an agent should react for a given set of circumstances.
- To fully monitor a network, packets from each segment should be monitored. This means that a packet monitoring utility (packet monitor) should be active on one PC per segment, but no more. However, the system should be able to

cater for the fact that a different PC on a given segment may need to take over monitoring duties.

- There should be an agent responsible for communicating with the network manager and then informing other agents of necessary details. The other agents should not need any human interaction in order to perform their duties.

For the above reasons it was decided to use three distinct agents, based on the work of Sycara et al. They came up with the idea of Information Agents, Task Agents and Interface Agents as follows for their RETSINA system:

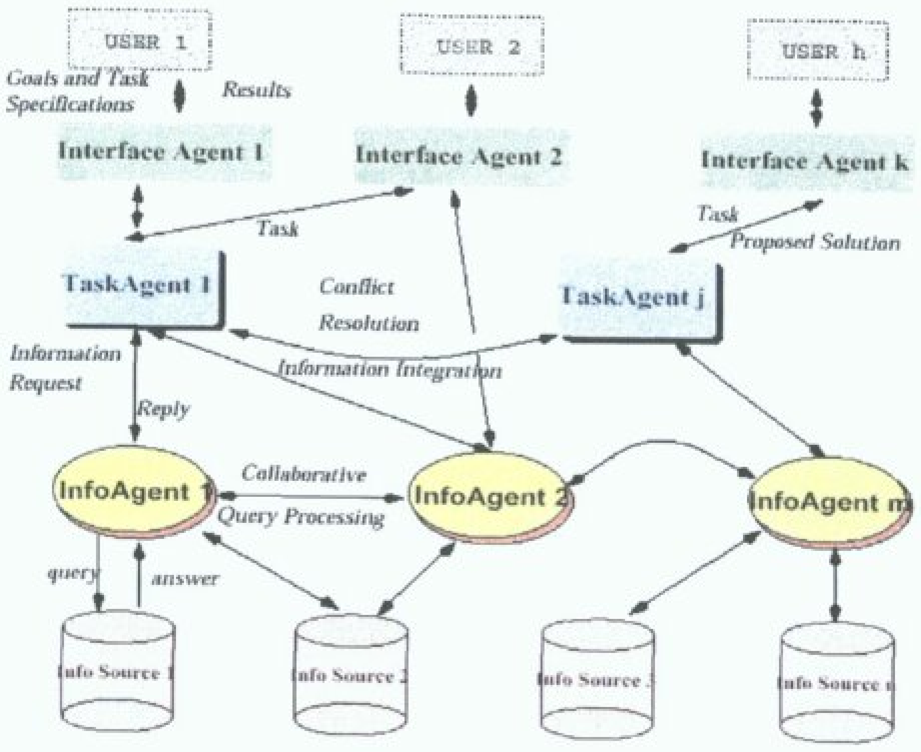


Fig. 3.3. RETSINA Distributed System Architecture

In the above example the Interface Agents are used to communicate with the end users' in order to achieve the users' goals. They are the only agents to deal with users and they will provide other agents with the information necessary to help achieve their goals. The job of the Task Agent is to formulate plans based on the users' goals and to communicate with other agents as necessary in order to exchange information and solve problems. The job of the Information Agent is to provide access to a heterogeneous collection of information sources. The architecture was adjusted to suit the network management paradigm as follows:

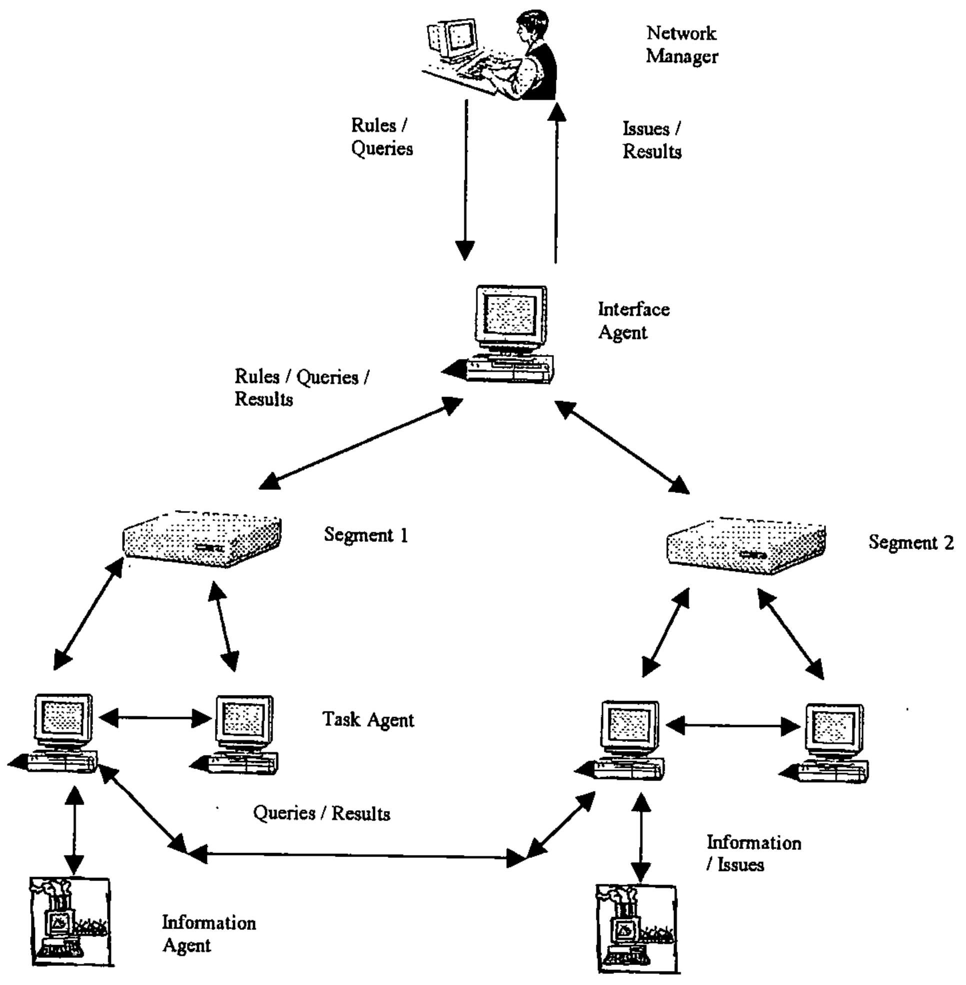


Fig. 3.4. Agent-based Network Management Architecture

The agents specified would have the following roles:

- The Interface Agent would be used to communicate with the network manager. There will be only one Interface Agent in the system. Its job is to report the status of the network to the manager when requested and also to report any serious problems and suggested solutions. It will communicate with the Task Agents but not with the Information Agents. It will pass on any new information to the Task Agents and also receive relevant segment information from them. This agent has overall control of the system and final say on any actions to be taken.
- The Task Agents will be located on each PC in the network. Their job is to negotiate among themselves until one Task Agent is in charge of managing each segment. They can communicate with each other and also with the Information Agent directly below them (i.e. on their PC). They will monitor the details being provided by the Information Agent and act appropriately. They will also monitor each other to ensure that if a Task Agent is no longer able to monitor its segment, another Task Agent can take over. Their main function is to recognise problems in the network and either solve those problems directly, with inter-agent communication, or by informing the Interface Agent.
- The job of the Information Agent is to simply monitor each Ethernet frame passing through its segment. It will record appropriate details about its network segment and if needed can communicate with its immediate Task Agent. However, it cannot make decisions on its own and is based on a purely reactive architecture. It simply reacts to the information presented to it. An

Information Agent will also be located on every PC but only one Information Agent will be running on a segment at any one time. These agents will also have the ability to send out raw Ethernet packets to test network conditions, when requested to do so by the Task Agent.

The Information Agent runs over NDIS as illustrated by the following stack:

Information Agent
NDIS
Network Card Driver
Network Card

Table 3.1 Application Stack

3.2 The Agent Communication Language

3.2.1 FIPA ACL and KQML

In order for the agents to communicate with each other, a protocol must be used so that the agents have the ability to understand each other. There were essentially two choices, KQML or FIPA ACL. There are many similarities between KQML and FIPA. They are both high-level, message-orientated communication languages for information exchange independent of content syntax. They are also independent of any transport mechanism (e.g. TCP/IP, SMTP etc.). There are three layers in a message to be sent. The content layer bears the actual message in the program's own representation language. This can be any language, e.g. Prolog, Lisp etc. The communication layer describes the low-level communication parameters such as the identity of the sender and the recipient. The message layer consists of the message that one application would like to transmit to another. In KQML a performative will be provided which specifies the type of the message i.e. an assertion, a query or some other performative. The KQML list of performatives is not a closed one and can be added to as needed. Performatives are called Communicative Acts (CA) in FIPA. The syntax of both agent communication languages (ACL's) is similar to Lisp. The initial element is the performative or CA, followed by the arguments as follows [31]:

```
(ask-if :sender A  :receiver B  :language VB
            :ontology NetMan  :content "Segment Load" :reply-with "load")
```

Below is a list of some of the main KQML reserved performatives for sender S and recipient R.

Name	Meaning
Achieve	S wants R to make something of their environment
Advertise	S is particularly suited to processing a performative
Ask-about	S wants all relevant sentences in R's VKB*
Ask-all	S wants all of R's answers to a question
Ask-if	S wants to know if the sentence is in R's VKB
Ask-one	S wants one of R's answers to a question
Broadcast	S wants R to send a performative over all connections
Error	S considers R's sentence to be mal-formed
Forward	S wants R to route a performative
Insert	S asks R to add content to its VKB
Recommend-one	S wants the name of an agent who can respond to a performative
Reply	Communicates an expected reply
Tell	The sentence is S's VKB
Untell	The sentence in not in S's VKB

Table 3.2. Main KQML Performatives
***VKB = Virtual Knowledge Base**

FIPA's ACL also uses semantic language (SL), which is a quantified, multimodal logic with modal operators for beliefs, desires, uncertain beliefs and intentions. It is used to represent propositions, objects and actions. A detailed discussion is not needed here but it is mentioned as it is the main difference between FIPA ACL and KQML. It should be noted that both languages are still very much in their infancy. Although there is a lot of research into each, at the time of writing there are no commercial applications using FIPA ACL. KQML is used in some commercial applications [31], [32] and thus "real world" feedback has been obtained. However, there are still no two applications developed by different organisations whose agents are able to communicate properly using KQML. ACL's have also been largely ignored by most of the Internet community. It was decided to use KQML for the network management application because of its popularity and also its relative ease of use. Some new performatives needed to be created which will be discussed in detail in the next chapter.

3.3 Storing the Agents Knowledge

It was not envisaged that the agents would need a lot of learning ability as most of what they would need to know can be determined before they are designed, i.e. network problems and their possible solutions. There were two possible approaches to storing the agents' knowledge. Case Based Reasoning and Rule Based Reasoning.

3.3.1 Case Based Reasoning

Case Based Reasoning (CBR) is a form of problem solving where new problems are addressed by retrieving stored records or prior problem episodes and adapting their solutions to fit new situations. In most systems the case adaptation process is guided by fixed case adaptation rules [33]. There are 4 main processes in the CBR cycle [34]:

- **Retrieve** the most similar case or cases
- **Reuse** the information and knowledge in that case to solve the problem
- **Revise** the proposed solution
- **Retain** the parts of this experience likely to be useful for future problem solving

Cases are usually stored in "problem, Solution" pairs although this structure may vary. CBR supports incremental learning as a new solution is retained each time a problem has been solved. Advantages that CBR offer include that fact that extensive domain knowledge is not required and also solutions can be found very quickly if a similar previous case exists. However, there is no way to explain why a particular solution was reached. This is because the solution matched a previous case. Also, large cases can suffer from efficiency problems due to the number of cases that may need to be retrieved and compared.

3.3.2 Rule Based Reasoning

A Rule Based System (RBS) consists of a knowledge base of rules. These will usually be represented as

If <Condition> Then <Conclusion>

statements. In order to use Rule Based Reasoning (RBR) an expert knowledge of the domain is required and it can be quite easy to convert this knowledge into rules. There are two ways of searching using RBR. In one case, facts are known and a conclusion is required. In this case forward chaining is used. In the second case, a conclusion is known but needs to be verified. In this case backward chaining is used.

With RBR, the link between rules is weak and so rules can easily be added or deleted without damaging the overall system. Also, it offers very good performance in a limited domain. An entire rule base can be searched very quickly and explanation facilities can be included because the rules are based on human expertise. Disadvantages include the fact that knowledge obtained is very task dependant and may not be adaptable to a slightly different task. The rules are not able to adapt and if information is missing the RBS will not be able to fire its rules.

3.3.3 Choice Made

It was decided to use a RBR approach for the network management application for the following reasons:

- The author already has an expert knowledge of Ethernet networks and so it will be reasonably straightforward to convert this knowledge into rules

- Ethernet Local Area Networks (LAN's) are a limited domain whose potential problems are well known
- The speed of RBS's will suit the reactive architecture of the Information Agents
- It will be possible to provide explanations which will be of great use to the network manager
- It will be easy to add new rules or adjust existing ones to suit the network (i.e. two people may disagree on what an overloaded segment is)

Conclusion

The above provides a very good basis for developing the agent management system. The system should be fast enough to react to immediate problems but also because of the rule base, the system will be able to cater for expanding and changing goals when required. Although new KQML performatives will be required, this should not add to the overall complexity of the system.

Chapter 4

Implementing The System

Because of the segmented nature of most of today's networks, it is very important that the Task Agents can determine a way to discover which devices are on which segments. It is also important that due to the reactive nature of the Information Agents, they should be running on PC's which have the processing power to deal with a large number of frames per second, while dropping as few as possible.

4.1 System Configuration

4.1.1 Installing the Software

The Interface Agent only needed to be placed on a PC that was going to be used by the network manager. A copy of the Task Agent needed to be placed on every PC and it was set up so that as soon as the PC was switched on, the Task Agent would begin running in the background. There was no user interface provided with the Task Agent so the user would be oblivious to its existence. A copy of the Information Agent also needs to be on every PC on the network. However, the Task Agent on a particular PC will decide when the Information Agent needs to be run and the Task Agent also has the ability to close down the Information Agent located on its PC. Therefore the Information Agent will not run automatically when the PC is switched on.

4.1.2 Determining a PC's Monitoring Ability

Because of the reactive nature of the Information Agent, it is important that any PC's that are monitoring Ethernet frames drop as few frames as possible to allow for accurate monitoring of the system. The Information Agent must be able to read every Ethernet frame on the segment, record appropriate details and inform the Task Agent of problems when necessary. It maintains a buffer of 10 frames but if this buffer fills, frames would start to get dropped before they were processed. The main consideration a Task Agent needed to take into consideration when determining the Information Agents ability to monitor frames was the PC's processor speed. The general idea being the faster a PC's processor is, the less likely the information agent is to drop frames. However, many of the PC's have software running in memory from system start-up that would use up processor cycles. Also, the PC's frame monitoring ability would be reduced by a large amount if a user were logged on to the PC because of the number of applications the user may be running. Therefore when the Task Agent starts up it will use CPUMark ©, which is provided by PC Magazine Labs. This utility calculates the processors ability, taking into consideration existing memory and also any software running in the background. It is an industry recognised benchmarking standard that is not limited to any one brand of processor.
The higher the rating provided, the better the PC's processing ability. Below is a table illustrating the results obtained when testing on the most common PC's in the network.

Processor	Memory	Applications Running	CPUMark Result
450MHz PIII	96MB	No	33.9
450MHz PIII	96MB	Yes	33.4
366MHz PII	128MB	No	32.5
366MHz PII	128MB	Yes	32.2
300MHz PII	32MB	Yes	23.1

Table 4.1 CPUMark

Examples of applications running would include a virus scanner and various network administration utilities. However, as mentioned before, when a student is logged on and using the PC, the amount of available processor cycles can vary dramatically. It was therefore decided the place a large burden on any PC that had a user logged on. A figure of 5 was subtracted from the CPUMark figure to give the PC's overall monitoring ability. In the example of a 366MHz PII processor with normal background applications running, the adjusted figure would be 27.2. It should be noted that in most daily situations almost all lab segments on the network had at least one PC where no user was logged on.

4.1.3 Detecting Other Task Agents in the Segment

In order to determine which Information Agent in a segment is going to be the monitoring one, the Task Agents within a segment need to know about each other and also their monitoring ability. Because the structure of Ethernet is flat a PC will not know which segment it is on nor will it know what other PC's are on its segment. Knowing a PC's IP address will not provide any solution to this problem although they could be grouped into segments. Therefore, when a Task Agent wishes to send its monitoring ability to other Task Agents on its segment, it needs to send out an Ethernet frame with the appropriate details enclosed. The only details it needs to include are its monitoring ability and its IP address.

This leads to the problem of how to ensure that every PC on a Task Agents segment will receive the Task Agents frame without any other Task Agent on a different segment picking the frame up. This is essential to ensure that all segments are known and no two Information Agents will be monitoring any one segment.

It should be noted that although a Task Agent can initiate the sending of a frame and the data to be enclosed, it is the Information Agent that actually creates the frame and sends it out via the network card's drivers. Also, when a frame is received the Information Agent must read it and extract relevant details for the Task Agent. The first idea was to send an Ethernet frame out with the source and destination address both set to the MAC address of the PC that was sending the frame. The idea was that the frame would get sent out along its own segment but not to any other segment because the destination PC would exist on the current segment. This would allow all the PC's on the current segment to read the frame and they would thus know that the frame came from their segment. Because of the fact that Information Agents on each segment would be monitoring perhaps thousands of frames per minute, a way was needed to indicate to the Information Agent that another Information Agent had created the frame, and that it was not just some random Ethernet frame. It was decided to use the DSAP field and the SSAP field to indicate that the frame had originated from a Task Agent, and had been sent by an Information Agent.

It was decided to set the SSAP field to 08 Hex and the DSAP field to 09 Hex in order to indicate that the frame had been created by the network management system.

Although there are only 256 possibilities for an SSAP field, because the fields will contain different values there will be no possibility of the agent management software mistaking a random frame for one which was created by the agent management system. Other frames will almost always contain identical values in the SSAP and DSAP fields.

When the frame was sent out with the source and destination MAC address being set to the MAC address of the current PC, it was noted that none of the other PC's on the segment received this frame even though the NIC drivers indicated that it had in fact sent the frame. It was then discovered the NIC drivers would not send a frame out on the network if the destination address on the frame were the same as the NIC MAC address. They will simply indicate that the frame has been sent. Making up an arbitrary MAC address would also not work because of the fact that the frame might traverse other segments on the network and this would allow other Information Agents on different segments of the network to read the frame, and mistakenly believe that it came from a PC on their own segment. If this problem could not be solved it would mean that the Task Agents would not be able to determine who is on their segment and who is on different segments. The solution found was to use a frame with a destination address that was guaranteed to be invalid. This would mean that the frame would traverse the current segment but would not reach any other segments. It was decided to use the address **00:00:00:00:00:00** as it was easy to recognise and no other NIC uses this address. Once the frame reaches a switch it will be deleted, as no device on a network will ever contain a MAC address like the above.

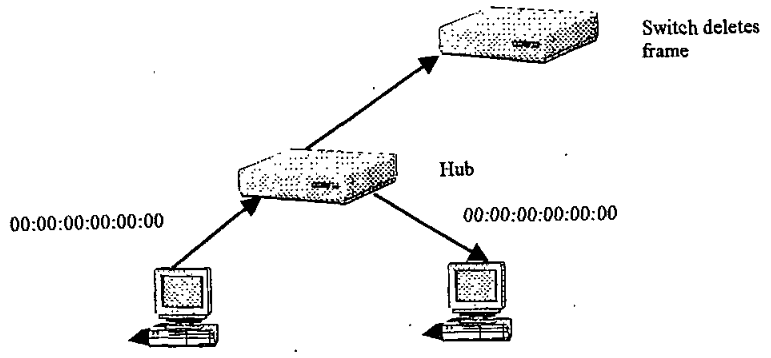


Fig. 4.1. Sending advertisement frame

4.1.4 Recording Other Task Agents

When an Information Agent is told by its Task Agent to send out the PC's frame monitoring ability it will create the frame as follows:

- The destination address will be set to 00:00:00:00:00:00 as mentioned previously
- The source address will be set to the PC's MAC address e.g. 00:80:C8:84:D2:F2
- The DSAP will be set to 09 and the SSAP field will be set to 08. This will use up the first two bytes of the data field
- The length field will be set to 46 as the data field does not need to be any larger but cannot be any smaller
- The next two bytes of the data field will contain the PC's monitoring ability as determined by CPUMark

- The rest of the data will contain the IP address of the sending Task Agent and a pad of zeros, in order to fill 46 bytes.

When an Information Agent receives a frame it will check if the DSAP and SSAP fields contain 09 and 08 respectively. If they do the Information Agent will know that it has received a frame from another Task Agent on it's segment. It will then extract the IP address of that Task Agent, along with its monitoring ability. It will then pass these up to the Task Agent via KQML. The Task Agent will then compare its own ability against the ability of the received frame. It will record the details of the other Task Agent in the Segments table of its Task Agents database. This table is used to keep a record of all the Task Agents on the current segment. If its own ability is better it will send a message to the other Task Agent specifying its ability. The other agent will also maintain a Segments table and will record the details. If the other Task Agent has a better ability then the local Task Agent will know that it cannot monitor the segment and so if its Information Agent is running it will stop it. It will obviously record the other Task Agents details in its segments table. When a Task Agent has sent its ability out it will wait for 10 seconds. After this time if it has the best ability it will start up its Information Agent so that it can monitor the segment. Whenever a new Task Agent starts up on the segment the process is repeated so that the most qualified Task Agent will be the one that is monitoring the network at any given time.

4.1.5 Recording Other Segments

Once a Task Agent is in charge of its own segment, it needs to inform Task Agents on other segments that it exists. Again, it will send out an Ethernet frame with its IP address but this time the destination address will be set to FF:FF:FF:FF:FF:FF, as this is a network broadcast address. When an Ethernet frame is sent with this address, every PC on the network will receive the frame. Again, the DSAP and SSAP portions of the frame will be set to 09 and 08 but because the frame has been set to a broadcast address, the Task Agent receiving the frame will know that it has come from another segment. Only Task Agents that have their Information Agents monitoring the network will receive the frame. Whenever one Task Agent takes over from another in network monitoring duties, it will inform the other segment Task Agents of the PC it is replacing so that they are kept up to date.

4.1.6 Detecting the Interface Agent

The Interface Agent can be run on any PC and does not need to be in memory all of the time, only when the network manager requires interaction with it. Therefore, the Task Agents need to know how to contact it and vice versa. This is because the Task Agents will inform the Interface Agents of any problems and solutions and the Interface Agents will also inform the Task Agents of new rules or solutions to existing problems. Therefore, when the Interface Agent starts up it will send out an Ethernet packet with a destination address of FF:FF:FF:FF:FF:FF. This will then be detected by all of the Information Agents that are monitoring the network. They will be able to detect the fact that it is from the Interface Agent because the DSAP field will be set to 07 and the SSAP field will be set to 08. The data field will also include the IP address of the Interface Agent so that the Task Agents can contact the Interface Agent directly. The rest of the data field will be padded with zeros, up to its length of 46 bytes. The Task Agents that are monitoring segments will then reply with their own IP address. The can also inform the Interface Agent of who is on their segment when requested and this will allow the network manager to see exactly how the network segments are set up.

4.2 Setting Up the Rule Base

4.2.1 Rules Used

As mentioned previously, it was decided to use RBR to deal with how the system handles problems. The network manager when using the Interface Agent specifies any changes to the existing rules. The Interface Agent then passes these changes on to the Task Agents that are monitoring their segments. The more specific the rules, the easier they are to maintain and change. One of the rules dealt with segment usage. It specified that if a segment was being 30% or more utilised (3 Mb/sec) then the network manager should be informed. A less used segment could then be suggested to place some of the PC's on if required. The rule was stored as follows:

```
If
        Average segment usage > 30%
Then
        Inform network manager
```

Again, the network manager could change the 30% figure to anything required. 30% load is given as the standard figure for which Ethernet networks will perform efficiently [35]. If the network manager wished to be informed if the segment usage was greater than 60% then the rule would be stored as follows:

```
If
        Average segment usage > 60%
Then
        Inform network manager
```

Another rule concerned what to do if a network segment went down. Obviously this is an extremely serious problem so the network manager would be informed immediately and also e-mailed, in case the Interface Agent was not running. The rule was stored as follows:

```
If
        Network segment down
Then
        Urgent inform network manager
```

The term "Urgent inform network manager" also relates to another rule. Again this helped the system more flexible because the network manager would then be able to choose how to be informed. The rules was stored as follows:

```
If
        Urgent inform network manager
Then
        e-mail network manager AND
        display on management station
```

The above rule specified that a message should be displayed on the network management station and the network manager should be e-mailed in the case of an urgent message. Another example might be to send a message to the network manager's mobile phone or beeper.

A further rule related to this specified that if the Interface Agent could not communicate with any Task Agent from a particular segment, then the segment was down. An explanation of how Task Agents discover that the monitoring Task Agent on their segment has gone down is discussed later. Another rule was used to determine when a PC was determined to be sending corrupt data. If the number of corrupt frames sent in a period of 5 minutes was greater than ten then it was determined as corrupt. Again, the number of corrupt frames or the duration could be changed. The rule was specified as follows:

```
If
        Number of corrupt frames > 10 AND
        Duration > 5
Then
        Corrupt application
```

A rule associated with this rule is to determine what to do when this happens. The rule specifies that the PC in question should be rebooted:

```
If
        Corrupt application
Then
        Reboot PC
```

Some of the rules used are listed in appendix A.

4.2.2 Explanation Facility for Rules

Because of the fact the RBR was used for storing the rules, it was possible to also store the explanation for those rules. These allow the network manager to fully understand why the system has come to a particular conclusion. The explanations for the rules were stored in a separate table with an associated rule number. For example, the above rule relating to segment overload would be stored as follows (assuming the rule number is 2):

2 The average segment usage (from 9am to 9:30pm) is greater than 30% of the maximum bandwidth of 10Mb/sec. This may increase network inefficiency due to excessive collisions, so it is recommended to move some of the PC's onto another less utilised segment.

If more than one rule was fired in order to reach a conclusion then the appropriate number of explanations would be available when requested. The explanations for the rules are listed in appendix B.

4.3 Agent Communication

As mentioned previously, all agent communication takes place via KQML. The messages were designed to be short and fairly infrequent so as not to add too much to the overall network load. However, new performatives were added to cater for specific messages relating to the network management software.

4.3.1 Advertising an Agent's Ability

When a Task Agent advertises its ability, it simply sends out an Ethernet frame so no KQML is involved. However, if an agent knows the location of all other agents on its segment and its ability has changed (perhaps a user has logged on or off) it will send a message to all other Task Agents on the segments specifying its new ability. To do this it will use the predefined KQML "Advertise" performative as follows:

```
(advertise :sender A :receiver B : language VB
        :ontology NetMan  :content "Ability (24.3)")
```

The above basically translates to

"sender A is sending a message to sender B. The message is that A's ability is now 24.3. The language being used by A is Visual Basic and the ontology is NetMan".

As no "Reply" performative was included in the message when receiver B get the message it will not reply to sender A. A and B refer to actual IP addresses such that A might be 193.1.116.253 and B might be 193.1.118.234. The ontology refers to the language in which the performatives were defined. NetMan is one that was created specifically for this system using some existing performatives and some which were created specifically for the agent system. This means that although another system would be able to break up the KQML message into its appropriate parts (i.e. the sender, the receiver etc.) it would not understand what "Ability 24.3" means unless it understood the performatives from the ontology NetMan. As mentioned previously this is one of the problems if inter-agent communication where different agents have been designed by different developers.

4.3.2 Querying Another Agent

When an agent wants to query another agent it must specify that it wishes to receive a reply. It can do this by using the "ask-if" performative. One example of this is where the Interface Agent wishes to determine the segment load from a particular Task Agent. The format would be as follows:

```
(ask-if :sender A  :receiver B  :language VB
        :ontology NetMan  :content "Segment Load" :reply-with "load")
```

The format is quite similar to the previous example except that sender A expects receiver B to reply with the label "load". This label will allow the Interface Agent to determine what the response from the sender is in relation to. When the Task Agent replies, it would reply with something like the following:

```
(tell  :sender B  :receiver A  :language VB
        :ontology NetMan  :content "30"  :in-reply-to "load")
```

As can bee seen from the above the Task Agent is informing the Interface Agent the segment is under an average 30% load.

4.3.3 Getting Another Agent To Perform A Task

Another situation occurs when one agent wants another to perform a specific task. An example of this would be where the Task Agent on a particular PC wants the Information Agent on that same PC to stop monitoring the network segment. In this case the "tell" performative can be used as follows:

```
(tell  :sender A  :receiver B  :language VB
          :ontology NetMan  :content "end" )
```

The receiver (in the above case the Information Agent) will understand the content "end" to mean that it should stop monitoring. However, the receiver will not know the reason for ending. One way to solve this problem would be to create a new performative called "end" and then allow the content to give the reason for the performative. The following KQML message would then be sent:

```
(end  :sender A  :receiver B  :language VB
          :ontology NetMan  :content "better(32.4)")
```

In the above example the Information Agent would know that it had to stop monitoring the segment and the reason was that a better Task Agent on another PC had developed an ability of 32.4 which was better that the ability of the PC which the Information Agent was on. It was decided to use this new performative to allow for more intelligence in the system and also because of the fact that it is the job of the Information Agent to record as much information as possible.

The same situation arises when changing the value of a rule in the rule base. The "tell" performative could be used but it was decided to use a new performative called "rule" to specify a change in the rule. The content of the message could then specify the rule to change as follows:

```
(rule  :sender A  :receiver B  :language VB
           :ontology NetMan  :content "change(13,25)"  :reply-with "changed")
```

The above message tells the receiver to change the condition for firing rule 13 (this rule happens to be the rule for determining when a segment is overloaded and had been set to 30%) from 30% to 25%. The sender also requests that the receiver respond to confirm the rule has been changed. This is very important to ensure that all Task Agents have up to date rule bases and that none are out of date. The receiver (Task Agent) could then respond to the sender (usually the Interface Agent) with the following to confirm the change:

```
(tell  :sender B  :receiver A  :language VB
           :ontology NetMan  :content "25"  :in-reply-to "changed")
```

This will confirm that the change has taken place.

4.3.4 Sending an Urgent Message to the Interface Agent

When the Task Agent needs to send an urgent message to the Interface Agent it can use the "urgent" performative as follows:

```
(urgent  :sender A  :receiver B  :language VB
           :ontology NetMan  :content "All Task Agents down"  :reply-with
           "urgent")
```

The above message informs the Interface Agent that all of the other Task Agents on the sender's segment appear to have gone down. Again, because of the importance of this message the sender expects an acknowledgement from the receiver. The acknowledgement would be as follows:

```
(tell  :sender B  :receiver A  :language VB
          :ontology NetMan  :content "received"  :in-reply-to "urgent")
```

The main agent messages are listed in appendix C.

4.4 Determining System Changes Needed

4.4.1 Changing the Monitoring Task Agent

It has previously been discussed how when the system starts up the Task Agents will decide who among them is best able to monitor the segment with their Information Agent. However, changes in the networking environment can take place and need to be catered for. If a new Task Agent starts up or detects a change in its monitoring ability (a user has logged on or off) then that Task Agent will inform all of the other Task Agents on its segment of its ability. These Task Agents will record the details and if the new Task Agent now has the best ability the previously best Task Agent will tell its Information Agent to stop monitoring the network and the new Task Agent will take over. There is also the possibility that a Task Agent that is responsible for monitoring a segment has gone down. This may be due to a system crash or because a user has switched off the PC. In either case the Task Agent will not have time to inform the other Task Agents about what has happened. This problem is solved by the fact that in any segment the monitoring Task Agent will send an "alive" message to the other Task Agents on the segment every 2 minutes as follows:

```
(tell  :sender A  :receiver B  :language VB
          :ontology NetMan  :content "alive")
```

If 2 minutes have passed and the other Task Agents have not heard from the Task Agent, the next best Task Agent will try to contact it. If there is still no reply the next best Task Agent will inform its Information Agent to begin monitoring the network and the other Task Agents will be informed. The monitoring Task Agent will then begin sending "alive" messages to the other Task Agents on its segment every 2 minutes.

4.4.2 Determining an Agents Ability

If a user either logs on or off a PC, the Task Agent will again run CPUMark to determine its ability. If its ability has changed its new ability will be broadcast as previously mentioned. Other than that, every week a Task Agent will use CPUMark to determine its ability. This caters for the fact that new software may be installed or removed, or new applications may be set up to run in memory all of the time.

Conclusion

The system as implemented now represents a very efficient way to monitor the network. KQML provides a flexible mechanism for agent communication and the rule base can also be changed at the network manager's discretion. Although there is a certain amount of work in setting up the agent software on every PC, once the software is installed it will only need to be configured via the Interface Agent.

Chapter 5

Comparison With an SNMP Application

In order to truly test the ability of the agent management software it is required to compare it with an existing, popular, fully functional network management system based on a popular protocol. A proper test should consist of instigating standard network problems and seeing how both applications perform. Network utilisation also needs to be considered as most networks are built based on the projected network traffic that will be passing through various segments.

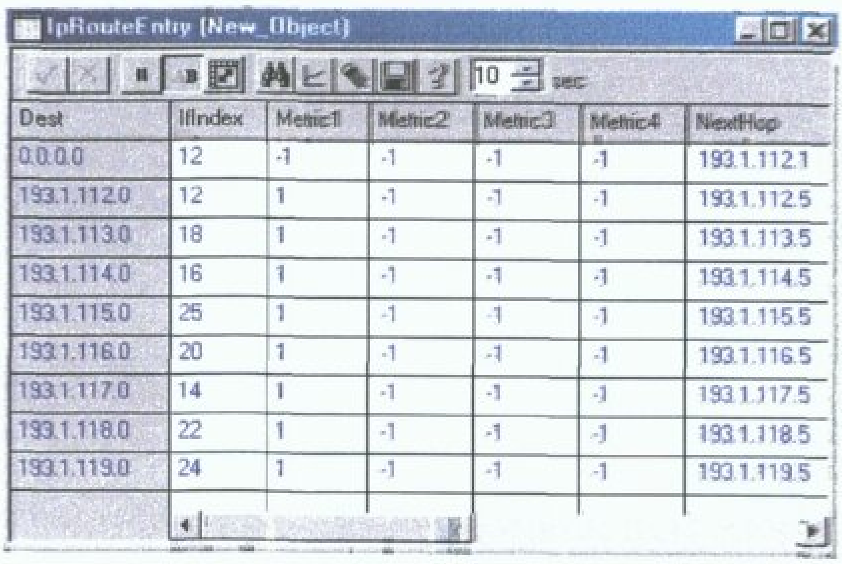
5.1 SNMPc

SNMPc is a network management tool provided by Castle Rock Computing. There have been a number of different versions of the product over the years and overall approximately 60,000 copies of the application have been sold, making it one of the more popular network management applications based on SNMP available today. The version used for comparison will be the Workgroup edition of SNMPc. This application is designed for small to medium sized networks of up to 1000 devices and is designed to be used by one person. Some of the main features of this software are:

- Runs under Windows 98 & NT
- Full MIB access
- Email/Pager Event Notification
- Advanced Event Actions
- Network Discovery with filters

- Real Time Tabular/Graphical Displays
- Device Specific Applications
- Printed Reports

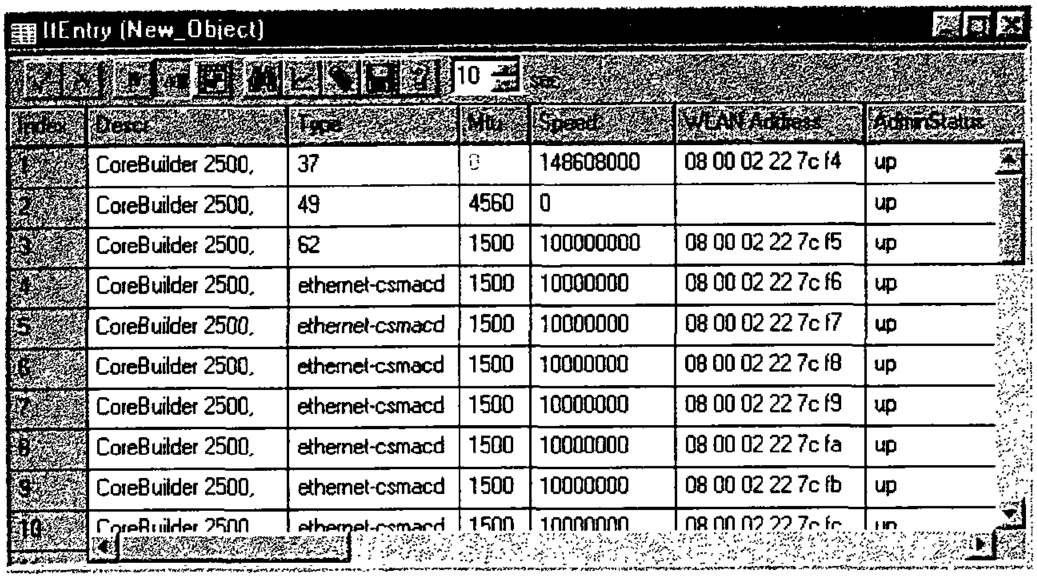
The main interface screen shows discovered segments on the network and all of the devices in each segment can also be displayed. SNMPc uses SNMP to communicate with switches and hubs and any other devices that support SNMP. The application allows full access to MIB's and can display packet and traffic statistics for any SNMP device. The application also allows a user to generate traps for conditions that require attention, for instance a port on a switch that stops transmitting and receiving data. Any devices that support it can also have settings on that device changed from SNMPc. Examples might include changing the status of a link or changing a devices routing table.


IpRouteEntry (New_Object)

10 sec

Dest	IfIndex	Metric1	Metric2	Metric3	Metric4	NextHop
0.0.0.0	12	-1	-1	-1	-1	193.1.112.1
193.1.112.0	12	1	-1	-1	-1	193.1.112.5
193.1.113.0	18	1	-1	-1	-1	193.1.113.5
193.1.114.0	16	1	-1	-1	-1	193.1.114.5
193.1.115.0	25	1	-1	-1	-1	193.1.115.5
193.1.116.0	20	1	-1	-1	-1	193.1.116.5
193.1.117.0	14	1	-1	-1	-1	193.1.117.5
193.1.118.0	22	1	-1	-1	-1	193.1.118.5
193.1.119.0	24	1	-1	-1	-1	193.1.119.5

Fig 5.1. Routing Table settings can be adjusted


IfEntry (New_Object)

Index	Descr	Type	Mtu	Speed	WLAN Address	AdminStatus
1	CoreBuilder 2500,	37	0	148608000	08 00 02 22 7c f4	up
2	CoreBuilder 2500,	49	4560	0		up
3	CoreBuilder 2500,	62	1500	100000000	08 00 02 22 7c f5	up
4	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c f6	up
5	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c f7	up
6	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c f8	up
7	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c f9	up
8	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c fa	up
9	CoreBuilder 2500,	ethernet-csmacd	1500	10000000	08 00 02 22 7c fb	up
10	CoreBuilder 2500	ethernet-csmacd	1500	10000000	08 00 02 22 7c fc	up

Fig 5.2. Port status can be changed

There is also a utility called HubView that allows a user to be given an exact graphical representation of a hub or a switch. This allows a user to view the type of ports on the device and also which ports are being used. Full statistical data can also be received from any port selected.

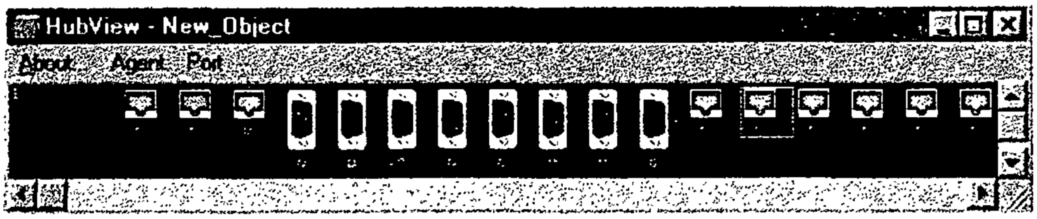


Fig 5.3. HubView

5.1.1 Initial Installation

As specified previously, the Agent-based network management software needs to be installed on every PC on the network in order to be able to determine which PC's are on which segments. After it is installed it can configure itself as needed and the network manager only ever needs to interact with the Interface Agent, which can be based on any station.

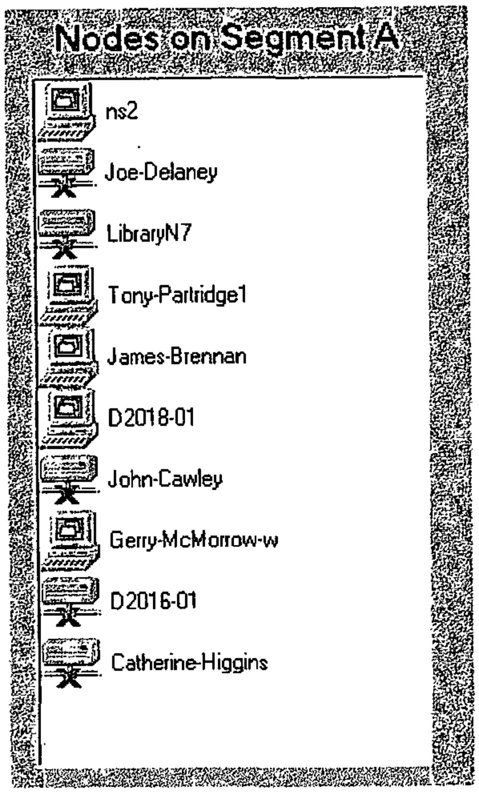


Fig 5.4. Agent-based view of a segment

SNMPc only needs to be installed on a single PC. Once it is started for the first time, it immediately begins talking to all switches and routers on the network in order to build up a picture of the network. It requires no user interaction to do this and takes only two or three minutes. It will then list all of the segments on the network and by selecting a particular segment the user can see what devices are contained in that segment. However, a problem can occur if complete automatic discovery is turned on. It will then try and connect to the complete network, which in the case of the network being tested meant the Higher Education Network (HEANet), which covers a number of colleges throughout Ireland. This can then set off intrusion detection alarms when SNMPc tries to poll some of these devices as the devices detect the poll as an attempt to gain unauthorised access to the network. Simply switching off complete automatic discovery can easily get around this problem. The user also has the option of setting the discovery agent so that only certain segments of the network (specified by IP filters) will be discovered.

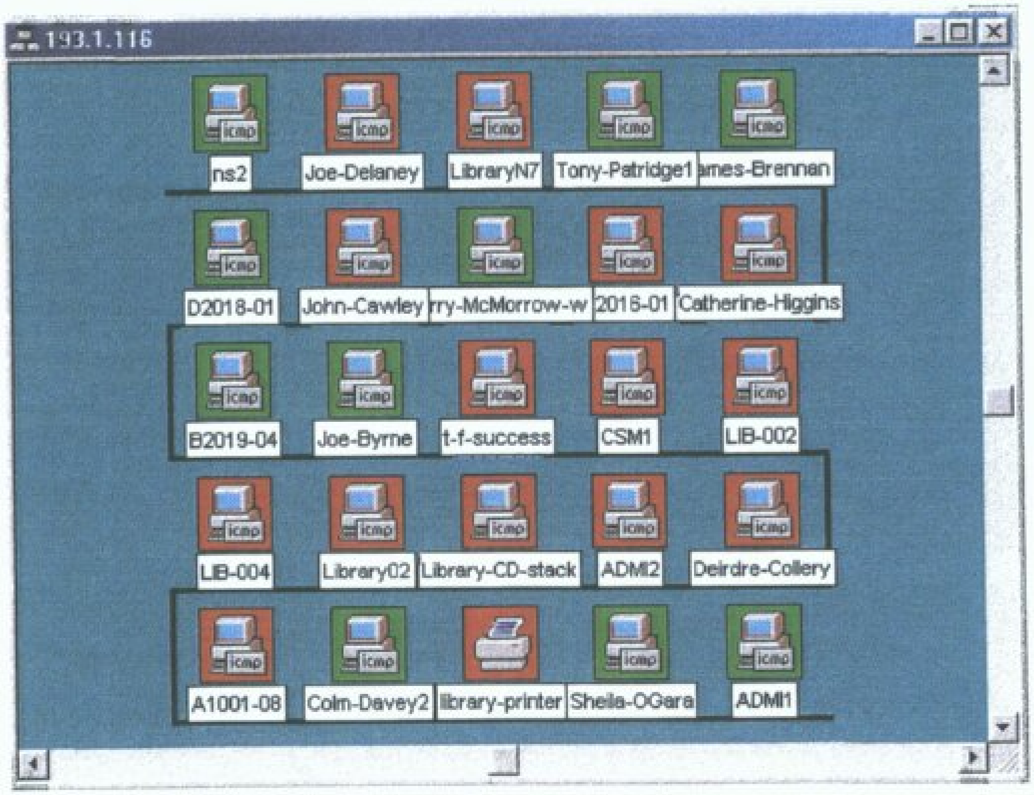


Fig 5.5. SNMPc view of a network segment

As can be seen from the above figures, some of the devices are currently not contactable and so have been displayed as offline. SNMPc can detect every device on the network, including printers. The agent management software can only contact devices which have the Task Agent installed on them.

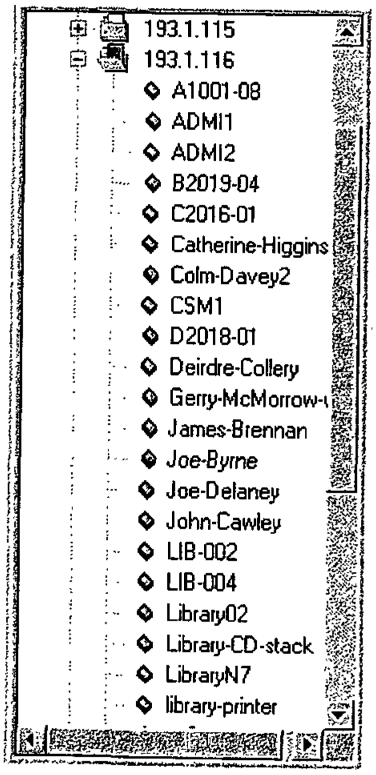


Fig 5.6. SNMPc list view of a segment

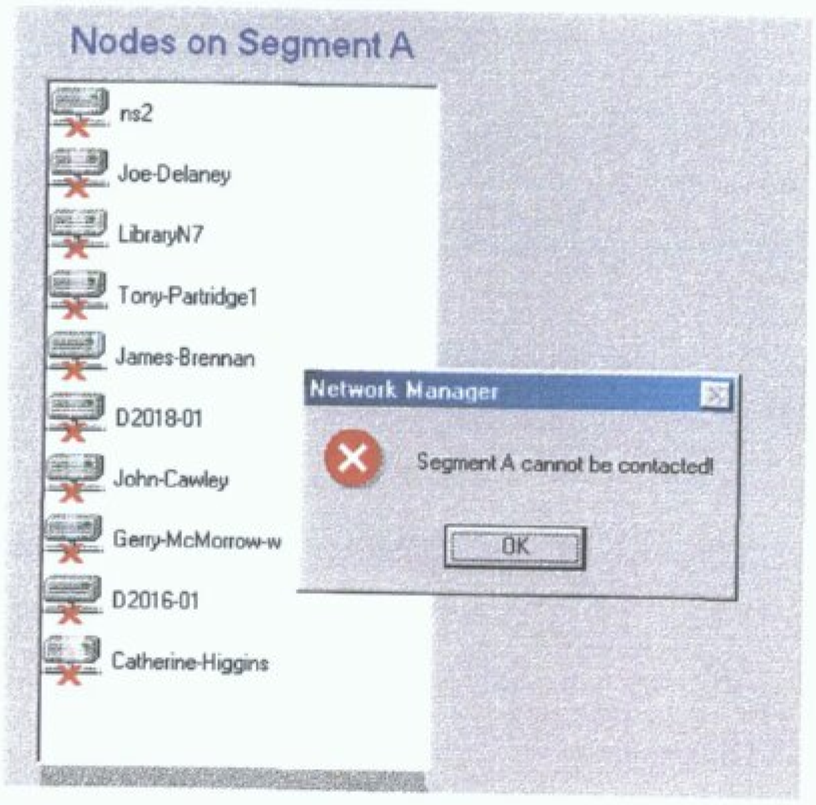
5.1.2 Detecting that a node has gone down

SNMPc can only "talk" to SNMP devices and so in order to detect PC's it simply sends them a ping every two minutes (this is adjustable). If it does not receive a reply, the node will be displayed in red on the user interface and a message will be displayed specifying the computer name of the computer that is no longer contactable.

With the agent management software, the monitoring Task Agent on each segment will send an “alive” message to all other Task Agents on its segment every two minutes (adjustable). The other Task Agents will then respond with an “alive” message of their own. If a particular Task Agent does not respond it will be recorded as having gone down but the network manager will not be informed unless the information is specifically requested. This is simply due to the fact that PC’s will often get switched off during the course of the day. If the Task Agent that is monitoring the segment goes down, this is a more serious problem. The other Task Agents will detect this due to the fact that they do not receive an “alive” message and the Task Agent with the next best ability will take over. The Interface Agent will be informed but again the network manager will not be informed unless the information is specifically requested.

5.1.3 Detecting that a segment has gone down

With the agent-based software, the Interface Agent will be able to inform the network manager within a couple of minutes if a segment has gone down. It will display a message on the station and also e-mail the network manager. The Interface Agent will receive an “alive” message every minute by each Task Agent that is monitoring each segment of the network. If it does not receive the message after a minute has passed it will know that either the Task Agent has gone down or the segment has gone down. If the Task Agent has gone down on a particular segment, the other Task Agents on that segment will detect the problem as discussed previously. The next best Task Agent will then begin monitoring the segment and will inform the Interface Agent. The Interface Agent will thus know that the segment is ok. If the Interface Agent receives no reply from any Task Agent on that segment after three minutes it will know that either the segment is no longer able to communicate with the rest of the network, or all of the PC's on that segment have been switched off. In reality, it is very unlikely all of the PC's on a large segment would be switched off so the Interface Agent knows that there is a problem with the segment. This most likely means that either a hub is not working or a connection from a switch to a hub has gone down. Either way the network manager will need to physically check the hub and/or switch in question in case it is a hub problem, a switch problem or a cabling problem. The Interface Agent on segments with just 1 PC will suggest no action unless the device has been down for more than 1 week. Another possibility is that the segment on which the Interface Agent resides is no longer able to communicate with the rest of the network. If this is the case, the Interface Agent will detect the fact that it cannot communicate with any Task Agents other than ones on its own segment and so it will realise that the problem lies on its own segment and it will inform the network manager accordingly.



Segment A cannot be contacted!

Fig 5.7. The Agent-based system displaying a failed segment

Because SNMPc can communicate directly with the switches on the network but not the hubs, it will be able to find out if there is a problem on the switch. If the link on a switch has gone down, the switch will record this fact and SNMPc will be able to display which link has gone down and generate an alarm if one has been specified. If a hub has gone down SNMPc will have no way to directly detect that there is a problem so the network manager will have to physically check the device. The same applies if a link on a hub has gone down. SNMPc can set alarms for switch problems (i.e. if the status of a switch changes) and this would consist of a message being displayed on the user interface and also an e-mail being sent to the managers e-mail address.

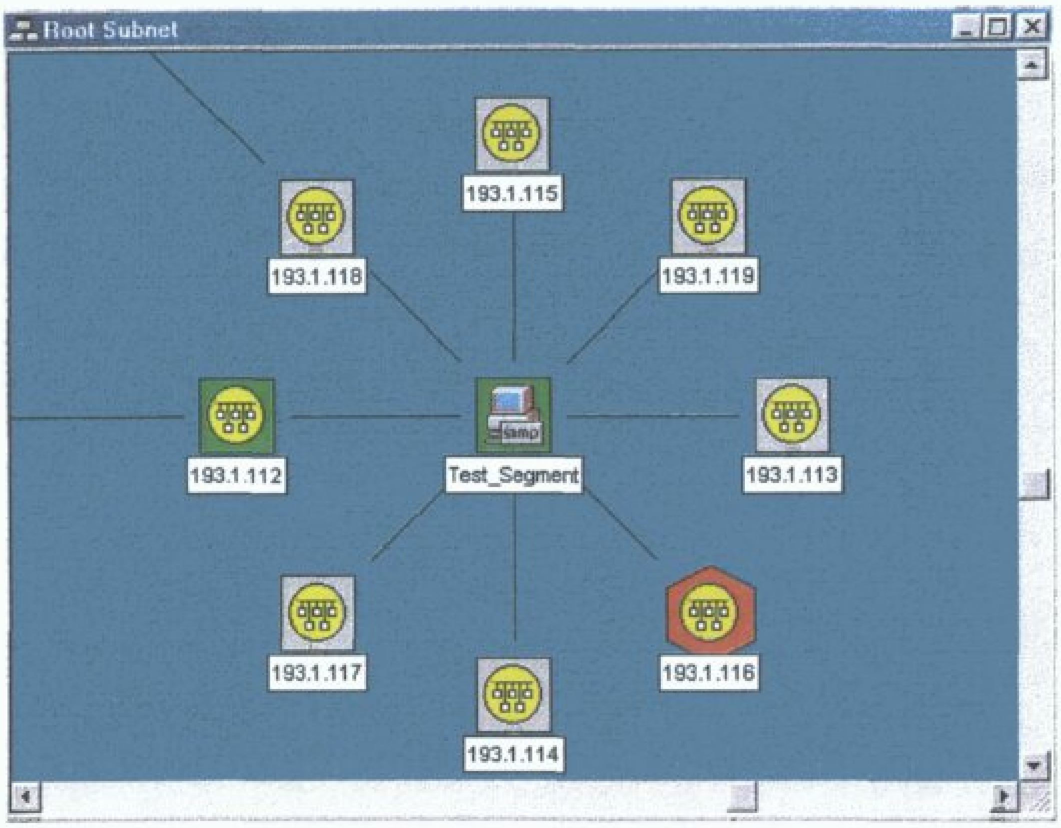
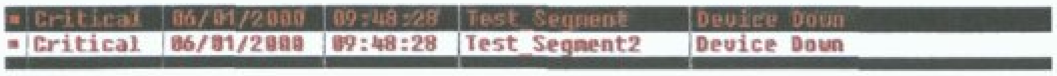


Fig 5.8. SNMPc segment before being brought down



```
▪|Critical  |06/01/2000 |09:48:28 |Test_Segment    |Device Down
▪|Critical  |06/01/2000 |09:48:28 |Test_Segment2   |Device Down
```

Fig 5.9. Alarm generated after segment went down

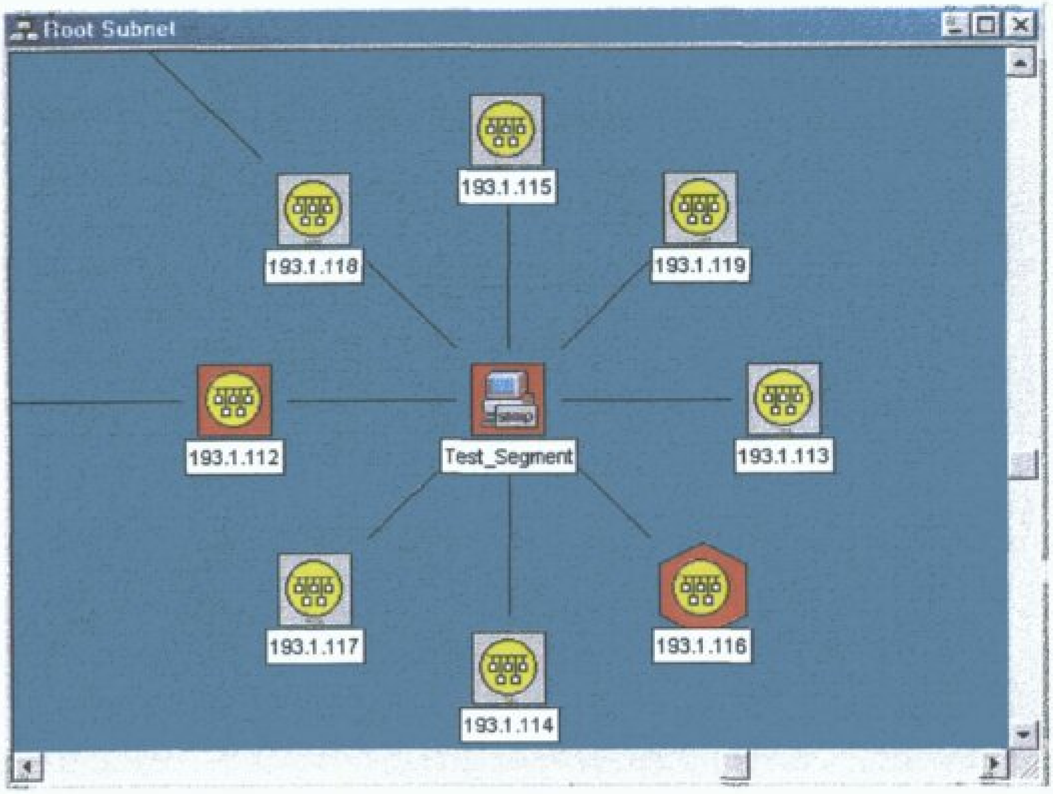


Fig 5.10. SNMPc display after segment went down

5.1.4 Detecting Network Utilisation

The chances of network failures in switches and hubs are very small, as are the chances of corrupt packets making it on to the network. One of the main uses of network management software is in showing network utilisation. Network managers typically want to know how much traffic their network segments are dealing with, if any are being overloaded and also what type of network traffic is traversing the network segments. These reports are also only of any real use if they can be generated over time, i.e. a number of days or even weeks. The agent management software is able to determine a number of facts about a segment over a given period of time and is also able to make suggestions based on the rules that have been provided to it.

5.1.4.1 Network Traffic

Network Traffic monitoring is concerned with how much traffic a segment is dealing with per second over a give period of time. All segments on the network are 10Mb/sec. This obviously means that if the traffic on a segment is approaching this figure over a sustained period, the likelihood of frame collisions increases dramatically. This in turn means that frames have to be sent again which increases the overall traffic on the segment and on the network, and the network will appear to be slower. The agent management software records the traffic traversing each segment over a continuous period. The Interface Agent will then allow the network manager to see the traffic over a given number of hours, days or even weeks if requested. It also should be noted that as with most networks, the traffic during the day is higher than at night. Therefore, only traffic between 09:00 and 21:30 was specified as being a day in the rule base, although a full day can be displayed if required by changing the rule base via the Interface Agent. The agent management software will generate an alarm for the network manager if the traffic on one segment is much higher than another that has been linked to it. By linked it is meant that during system set-up, the network manager has specified that a number of segments have been set up similarly and therefore should display the same kind of utilisation over any given period. This information is stored in the rule base. In the case of the test network, this applied to different labs, i.e. labs used for software development consisted of 32 PC's in each segment and so should be utilised quite heavily. Labs used for secretarial studies only had 16 PC's per segment and so should not be as heavily utilised. PC's belonging to administration staff only has between 1 and 5 PC's per segment and so could not be considered the same as the lab segments. If two similar segments are showing different traffic loads over a continuous period then the Task Agent in question will inform the Interface Agent, which will inform the network manager with a proposed solution based on the rules that were being fired. Another situation arises when a segment is under heavy use over a sustained period. The definition of heavy use and a sustained period are both stored in the rule base.

5.1.4.1.1 Segment Comparison

To test similar segments against each other, the Task Agents were left to monitor two segments continuously over a period of one week, during normal college term. The segments were assigned the names of *Segment A* and *Segment B*. The rule base was adjusted so that a sustained period was set to one week and that a noticeable difference between segment traffic was set to 15%. At the end of the period, it was noted that Segment A had an average load of 30.808 KB/Sec and Segment B had an average load of 12.1 KB/Sec over the given period.

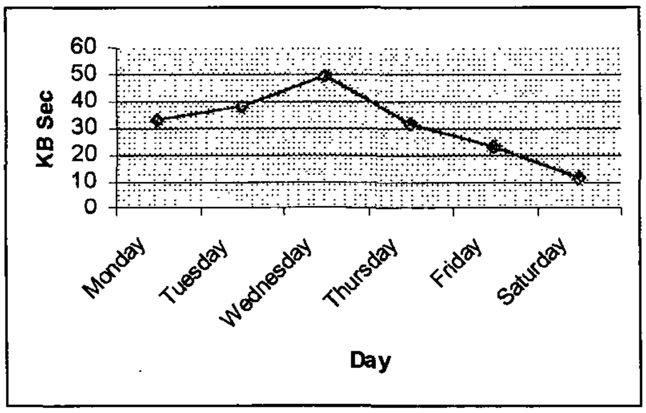


Fig 5.11. Overall traffic over 1 week on Segment A

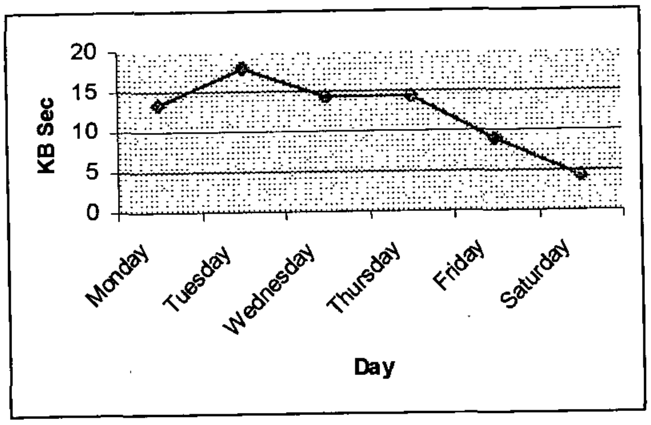


Fig 5.12. Overall traffic over 1 week on Segment B

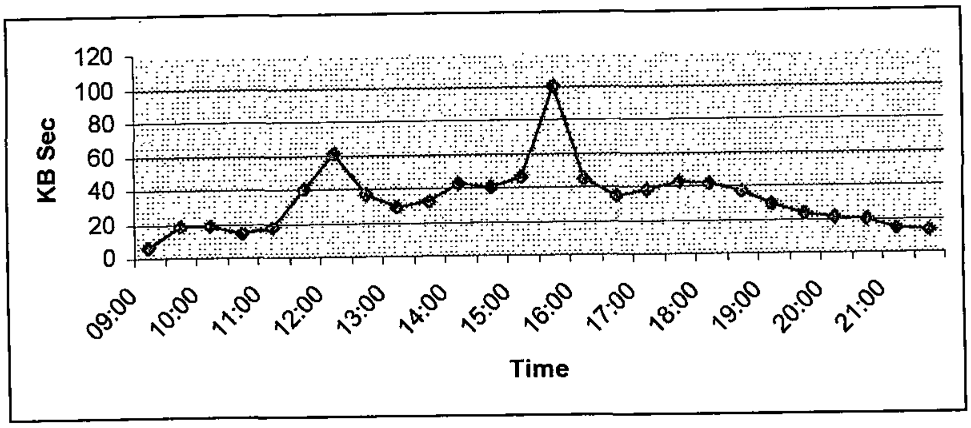


Fig 5.13. Segment A traffic over 1 day

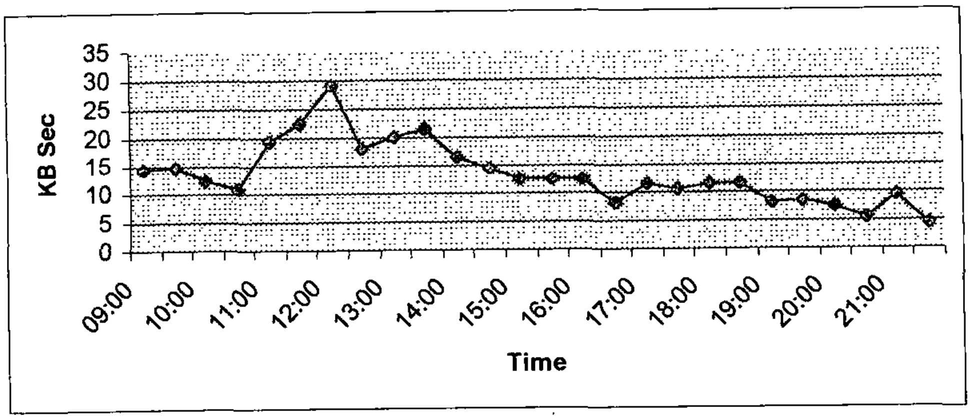


Fig 5.14. Segment B traffic over 1 day

When the type of traffic was analysed, it was determined the IP traffic was considerably more prevalent in Segment A than in Segment B. The Interface Agent therefore suggested to the network manager that some of the PC's in Segment A be swapped with some of the PC's in segment B, in order to even out the traffic.

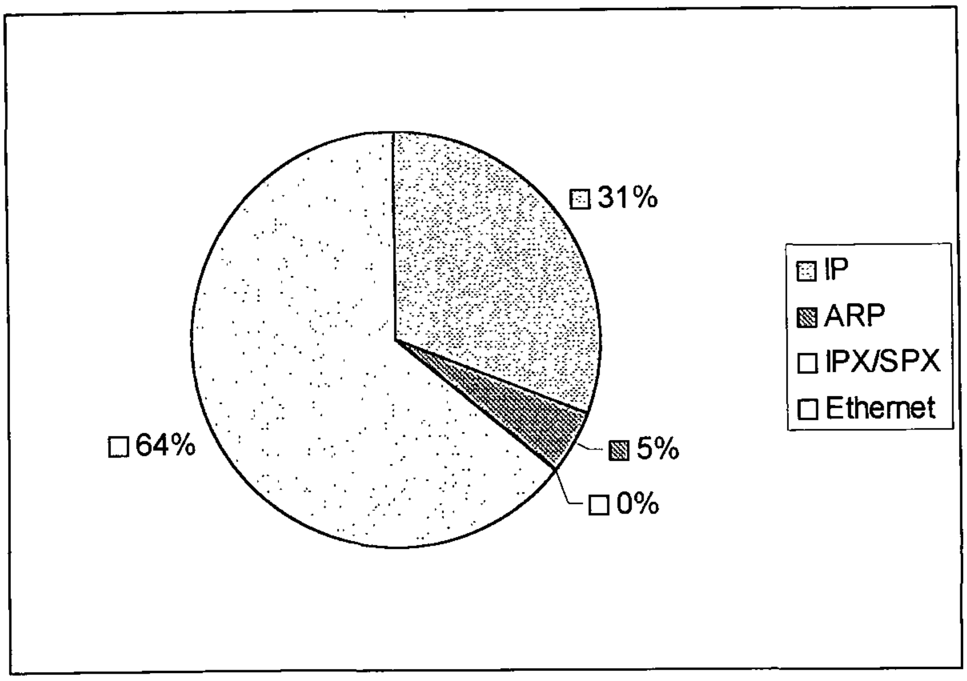


Fig 5.15. Breakdown of segment A traffic over 1 week

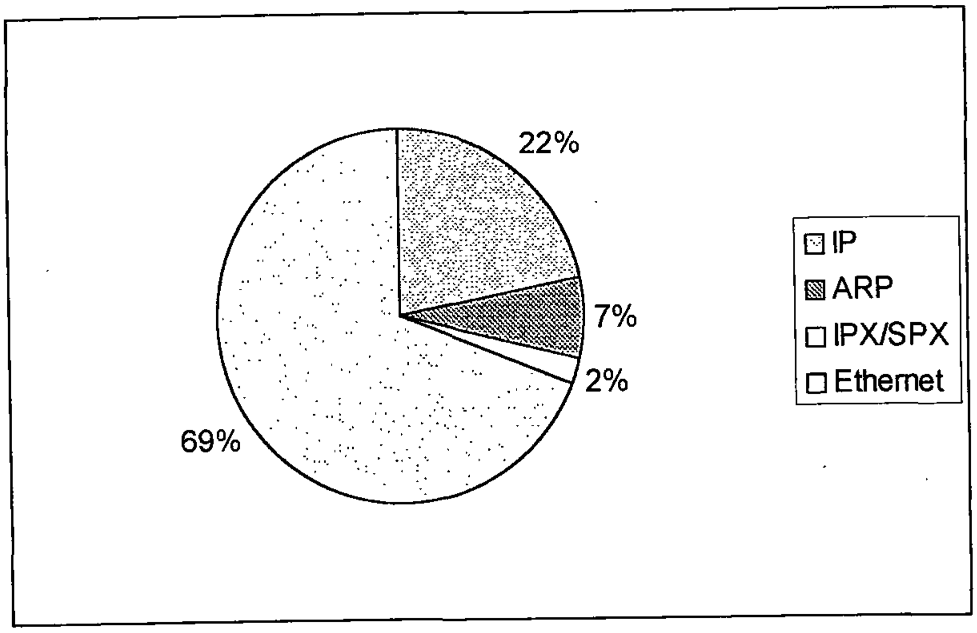


Fig 5.16. Breakdown of segment B traffic over 1 week

While SNMPc is able to show the amount of traffic generated over a given period it was not able to determine that the two segments were connected and was unable to suggest a solution to the test problem generated, as it did not see a problem occurring.

5.1.4.1.2 Segment Overload

Another test situation was set up to deal with segment overload. In this case, a continuous period was defined as one day and overload was determined to be 30% utilisation. Again two segments were used, Segment A which was going to be overloaded, and Segment B which was not. Over the period of the day, software was placed on a number of PC's on segment A that would continuously download files from an FTP server on a different segment. After the time period had passed Segment had an average load of 31.56% of total available bandwidth and Segment B had an average load of 1.205% of total bandwidth.

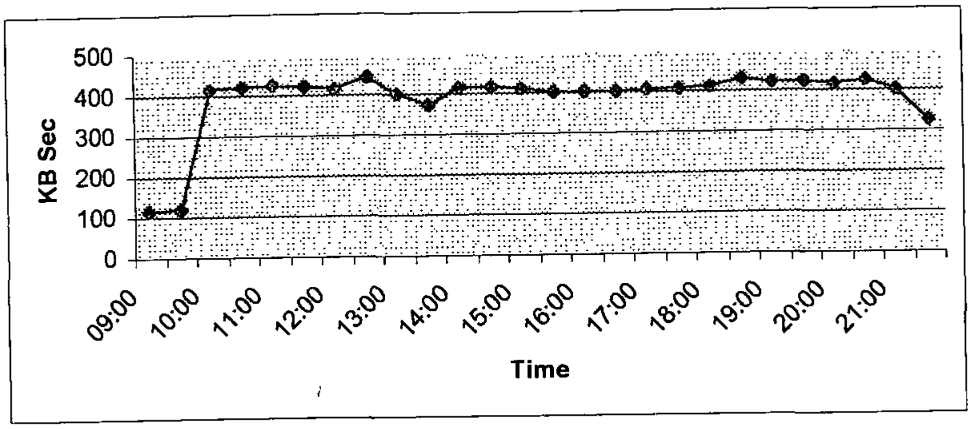


Fig 5.17. Segment A being overloaded

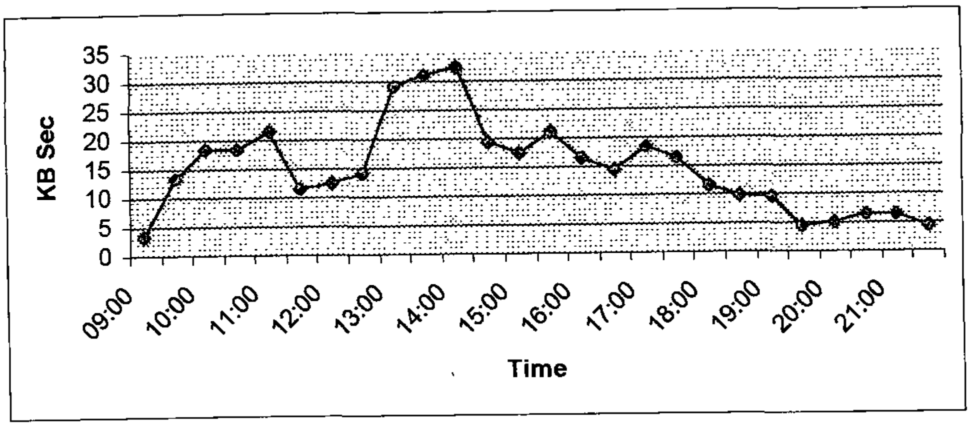


Fig 5.18. Segment B being utilised normally

The Interface Agent informed the network manager and again suggested that some of the PC's in question (only the ones that were downloading heavy amounts of data) be placed on Segment B. It did this because the first rule to be fired was that if the traffic on a segment was greater than 30% of the bandwidth, the segment could be considered overloaded. This can happen during normal events that include lunch hour, when most people in the labs are surfing the Internet, and also on the top of every hour when students log into the server. However, the network manager will not be informed until an appropriate period has passed. In this case the continuous period had been set to one day. SNMPc is able to generate an alarm if traffic exceeds specified limits but it will not take into account peak times (e.g. lunch hour). Therefore, a large number of alarms may be generated in a single day, many of which are not relevant.

5.1.5 Collisions

The agent-based management system has no way to detect collisions on a link. The NIC's drivers will detect a collision and will prevent more frames from going out on the link until the link is clear again. However, the drivers will not tell any higher-level applications, including NDIS which is what the Information Agents' frame monitor uses. In order to solve this problem, it would have been necessary to develop a new set of drivers for the NIC's being used on the network. This is itself would be very difficult as it would involve getting the hardware specifications from the card's manufacturer but because of the large number of different NIC's on the network, it was deemed beyond the scope of the research to be carried out. The switches on the network however can detect collisions and these are recorded in the appropriate MIB entry. SNMPc shows the number of errors that have been received on a given segment or port but it does not specify which were the results of collisions.

5.1.6 Dropped Frames

Obviously monitoring a network is better when the number of frames that are dropped by the monitoring utility is as low as possible. A dropped frame can be thought of as one that the monitoring utility does not have time to process because the number of frames arriving at a given time is so large the monitors buffer fills before the packets can analysed. Dropped frames will most likely reach their appropriate application without any problem but if the agent-based management software cannot read them then they are considered as being dropped. To consider the number of frames that can be dropped let us first consider the number of frames that can pass through a link in 1 second.

An individual Ethernet frame can be between 72 and 1526 bytes in length. Therefore the frame rate varies considerably with frame size, the smaller the average frame size the more frames per second can pass through a link.

Ethernet operations require a dead time of 9.6 μs between transmissions of two frames. We will only consider 10Mbit Ethernet. Because 10 million bits can traverse an Ethernet link in 1 second, the bit time is 100 nanoseconds. Taking only 72 byte frames the maximum number of frames that can pass through a link is

9.6 μs + 72 bytes * 8 bits/byte * 100 ns/bits

⇨ A single frame can pass in 67.2 * 10^{-6} s or

⇨ 14,880 frames can pass through in one second

A two PC segment that had no other traffic other than one PC sending frames that were 72 bytes in length was then used to test the number of frames that would be dropped. The other PC had the agent management software. A burst of 100,000 frames was sent out on the segment. At the end of the test, the receiving PC had been unable to process 160 frames out of the 100,000 sent which represented a drop rate of .16%. This is quite a small drop rate however the testing conditions were not very comparable with normal PC use over a multiple PC segment on the network. The next test was carried out on 2 PC's in one segment. One of the PC's had no user logged on; one of the PC's had a user logged on performing normal computing operations. The monitoring software ran for a period of 4 hours during normal lab use. At the end of the 4 hours the average drop rate for users logged on was 0.0022% of frames received while the average drop rate for users logged on was .0085%.

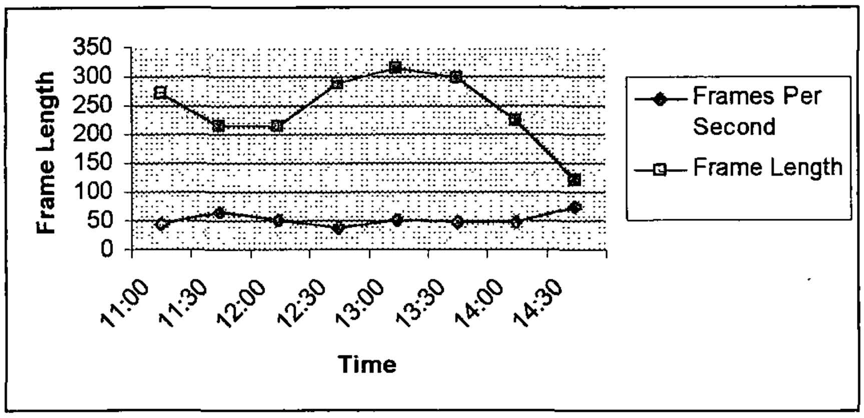


Fig 5.19. Traffic and frame lengths with a user logged on

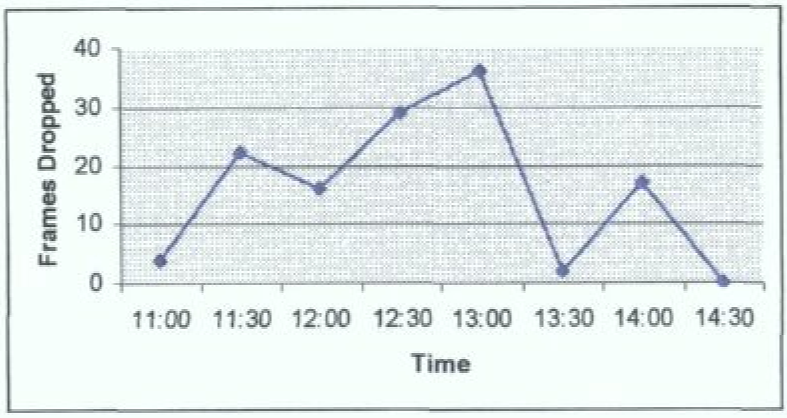


Fig 5.20. Number of frames dropped on the same PC

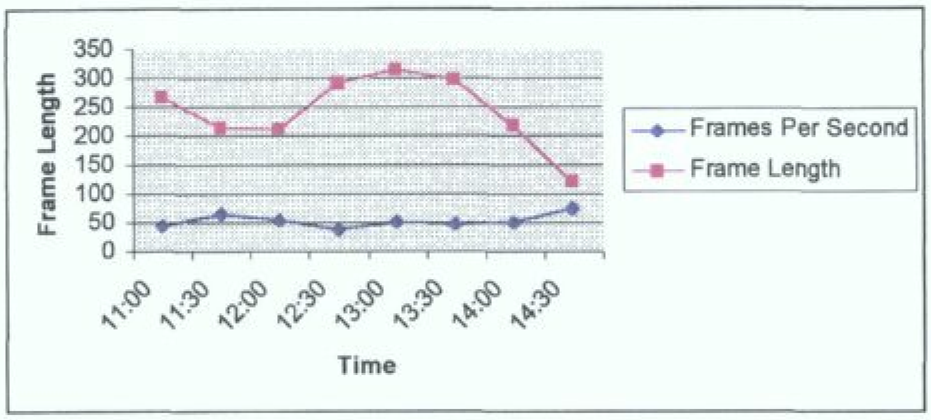


Fig 5.21. Traffic and frame lengths without a user logged on

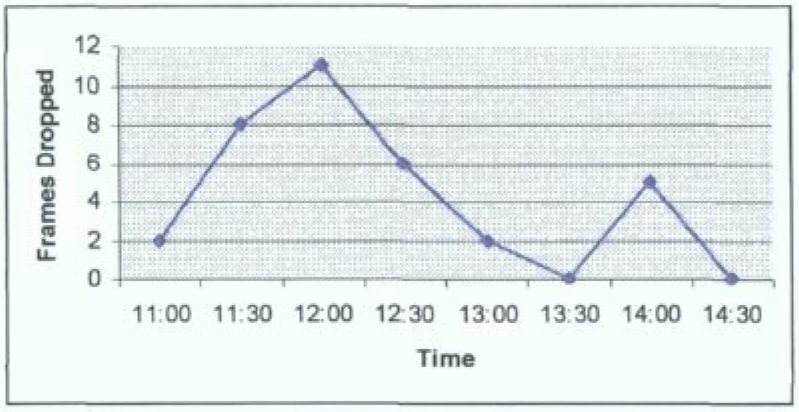


Fig 5.22. Number of frames dropped on the same PC

This indicates that the problem of dropped frames is not serious enough to seriously corrupt long-term results such as segment usage and details of corrupt frames.
SNMPc does not have this problem as the devices that support SNMP have the ability to process all of the frames that they receive.

5.1.7 Sending corrupt frames onto the segment

The final test to be carried out concerned sending a burst of corrupt frames onto a segment to see how both network management applications would react. The test was set up by having a piece of software send out a burst of 50 corrupt frames over a period of one second. There were two types of corrupt frames considered. The first type of corrupt frame is where the frame is physically corrupt. This can be obtained by sending a frame that is too large, too small or sending a frame where the result of the Frame Check Sequence (FCS) field is incorrect.
Sending this type of frame requires special software which was obtained from 3com. Normally, a NIC's drivers will not allow such a frame to be sent out but if the drivers become corrupt it is possible for corrupt data to be sent out on the link. It should also be noted that these type of frames will only traverse one segment, once they reach a port or hub they will either be deleted, or in the case of jabber (a frame which is larger than 1576 bytes), they will be cut down to the maximum size and sent on to the appropriate destination. The second type of corrupt frame is one in which the frame size is correct, as is the FCS field, but the frame contains invalid data. An example of invalid data would be a length field that was zero. These frames will traverse the appropriate number of segments until they reach their destination, where they will be most likely rejected by the host application that required the frame. The first type of

frame shall be considered *Frame Type A* and the second type of frame shall be considered *Frame Type B.*

5.1.7.1 Testing with Frame Type A

While using the network management application, a burst of 50 frames was transmitted on a 32 PC segment of the network. The frames had correct data but the FCS field was changed to contain an invalid value.

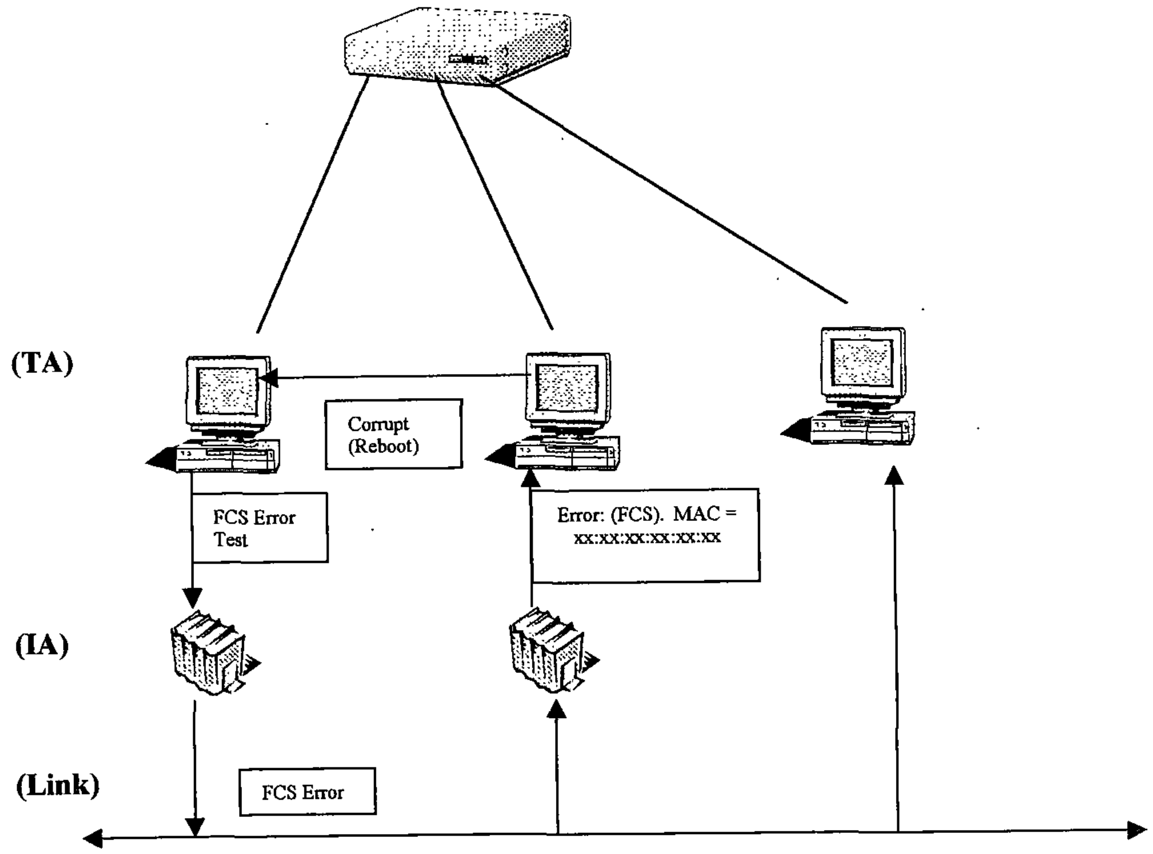
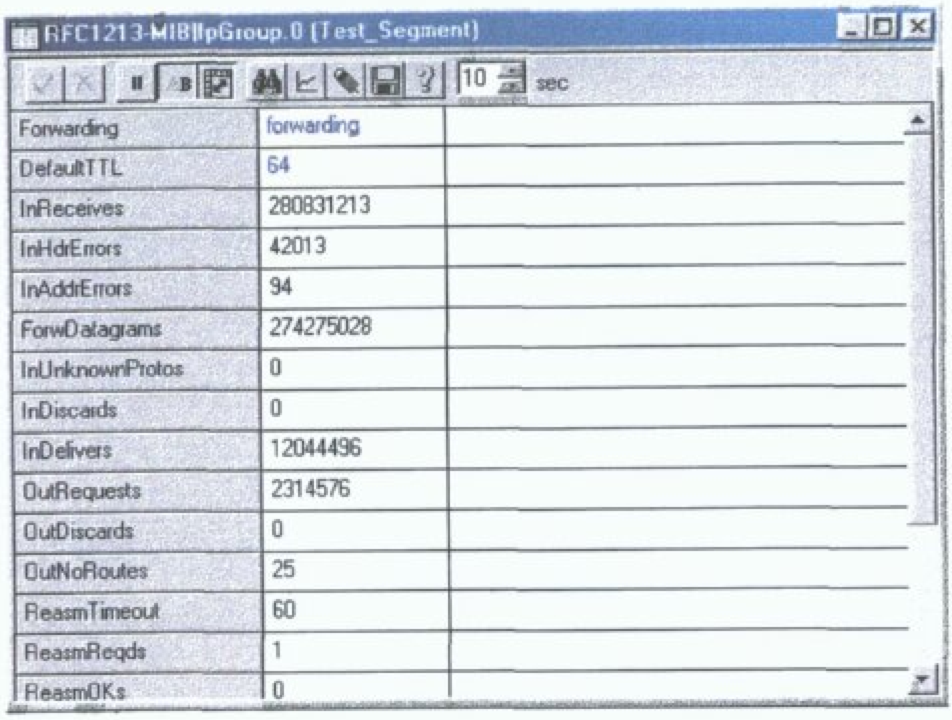


Fig 5.23. Sending a corrupt frame onto the segment

TA = Task Agent. IA = Information Agent

The Task Agent that was monitoring the segment received a "corrupt" message from the Information Agent every time a corrupt frame was received. This message contained the source and destination address of the frame and also the problem with the frame. The Task Agent then recorded the frames details and checked if a corrupt frame from the same destination address had arrived recently, within 5 minutes. The rule base had been set up so that if 10 corrupt frames were received from the same source address over a period of 5 minutes, the PC's NIC would be considered corrupt. After 10 frames were received, the Task Agent would check if the frame had arrived from a local PC. It would simply do this by checking the list of MAC address on its own segment. As mentioned previously, it should be guaranteed that the frame is local, as it will be deleted once it had reaches a switch or hub. The Task Agent will then contact the Task Agent on the PC that sent the frame with a "corrupt" message. The Task Agent that receives the message will then reboot the PC in question to see if the problem can be solved. The network manager will not be informed but can request the information. If after the PC reboots, the problem still exists, the network manager will be informed as there is nothing else that the network management software can do.

With SNMPc, when the switch received Frame Type A, it deleted the frame and added 1 to its number of corrupt frames received. It did not, however, take note of where the frame came from nor did it detect the fact that multiple corrupt frames had come from a particular PC.



RFC1213-MIB!IpGroup.0 (Test_Segment)

10 sec

Forwarding	forwarding
DefaultTTL	64
InReceives	280831213
InHdrErrors	42013
InAddrErrors	94
ForwDatagrams	274275028
InUnknownProtos	0
InDiscards	0
InDelivers	12044496
OutRequests	2314576
OutDiscards	0
OutNoRoutes	25
ReasmTimeout	60
ReasmReqds	1
ReasmOKs	0

Fig 5.24. SNMPc displaying corrupt frames

5.1.7.2 Testing With Frame Type B

A PC was set up in a standard 32 PC segment that would send out a burst of 50 packets with a length field of zero. The results for the agent-based network management application were the same as with the previous test. The Information Agent passed a "corrupt" message onto the Task Agent. Again, after the appropriate number of corrupt frames had arrived from a particular PC, the Task Agent checked the source MAC address and determined that the frame was on its own segment. It again sent a message to that Task Agent so that the PC in question was rebooted. However, because switches will not delete these frames, they are able to traverse multiple segments. This meant that other Information Agents on other segments (in the test case one other segment) would be able to detect the frame as being corrupt.

The other Task Agent will check the MAC address against its own list of MAC addresses and because the frame has originated on another segment it will not be able to find it. It will therefore send a "find" message to all other monitoring Task Agents on the system to see if any of them know the source MAC address. Each Task Agent will then check the MAC address against its own list of MAC addresses and the one that finds a match will return with a "found" message. If no Task Agent is able to locate the source address it will be determined that the frame must have originated on another network. In this case the network manager will be informed.

As SNMP devices do not analyse frames to check their actual content and there was not a problem with either the frame size or FCS field, SNMPc will never detect these types of frames and so no action will be taken.

5.1.8 Traffic Generated by the Network Management Systems

Because of the nature of both the agent-based and SNMP based network management systems, both systems are going to add to the overall traffic on the network. This is because they need to communicate with devices across the network in order to properly monitor the network. Traffic generated will only become a problem if it is putting the network under undue pressure, e.g. overloading a segment or creating too many collisions.

5.1.8.1 SNMPc

To test SNMPc, a segment was monitored for two hours during a period in which there were no users on a segment. The result of the test showed that the traffic traversing the segment was only 263 bytes per second. SNMPc was then started up and left to monitor the network in its standard set-up. This means that SNMPc would simply poll the devices every so often and was not generating any reports. After two hours it was noted that the traffic on the network had increased to 1,010 Bytes/sec. This was not a large increase and would be very unlikely to cause problems on a standard segment, even on a busy day. Finally, SNMPc was set-up to constantly monitor a port on a router. This meant that the software had to receive packet details from the router every 10 seconds and record the types of packets that were arriving at the router. This increased the traffic to 1,610 Bytes/sec, which meant that overall traffic had increased by approximately 1.4 KB/sec. This would not cause any problems for the test network even during busy periods.

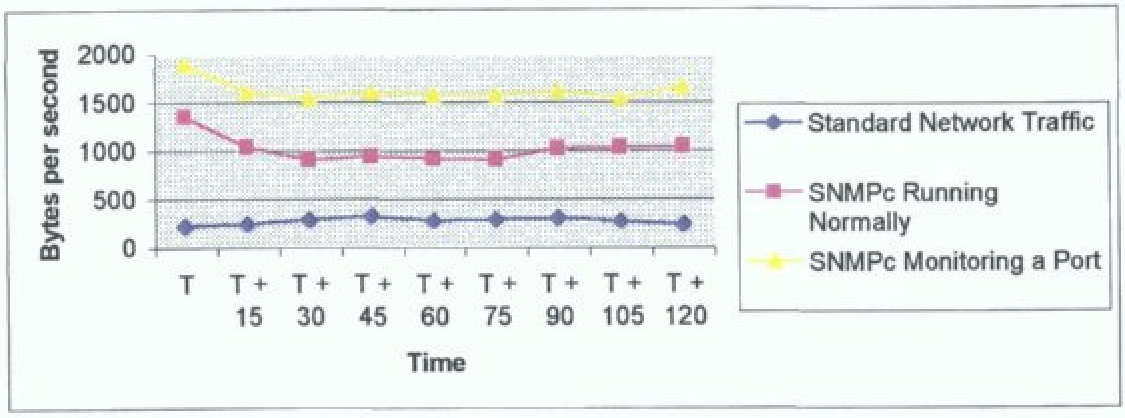


Fig 5.25. Traffic details with and without SNMPc running

5.1.8.2 The Agent-based Software

The agent-based software was tested using the same conditions on the same segment. After the first two hours the segment had a load of 284 bytes per second. The agent-based software was then started on the segment. Although the Task Agents on the segment initially send a few messages to each other, this stops after a couple of minutes and they resort to sending their standard "alive" messages. The agent software was left monitoring the network for two hours and after this period the traffic on the segment increased to just 312 bytes per second. It was therefore concluded that neither of the two applications would cause any problems with regard to traffic on the network. Although SNMPc generates more traffic, the station it would be located on in the test network would be an administration station; therefore very few PC's would be on that particular segment thus freeing up more bandwidth for SNMPc.

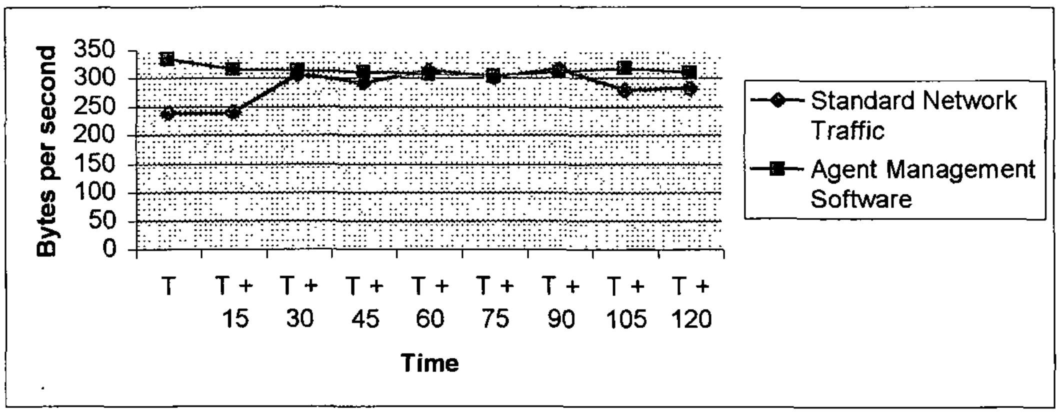


Fig 5.26. Traffic details with and without the agent software running

Conclusion

The above has been a thorough evaluation of both network management applications based on common errors that can occur in LAN's. Two of the more surprising results that were discovered were the fact the neither application adds to overall network traffic by any large degree and also the network itself is utilised very lightly, therefore problems that occur are rare. The fact that very few frames are being dropped by the agent-based application is also a bonus. However, in a switched environment where every PC was attached to the port of a switch there would be problems with the agent based management system. An Information Agent would need to be active on each device and this would add quite a lot of extra traffic. Also, the system would slow down every PC on the network and a PC would need to have the Information and Task Agent active all of the time, regardless of the PC's processing power. It should be noted that very few networks currently operate in a totally switched environment.

Chapter 6

Conclusions

Now that a complete examination has been carried of the agent-based network management system, a number of conclusions can be drawn.
An excellent knowledge has been obtained on agent architectures and the one chosen (based on RETSINA) was very beneficial to the agent-based network management system. The Information Agent had a near 100% frame-processing rate and was able to read the complete contents of a frame because of its reactive architecture, but was also able to pass on relevant information in whatever format was required by the Task Agent. It would be very straightforward to change the Information Agent to extract different types of information, without changing the other agents. For instance, the Information Agent could be used solely as a protocol analyser. The main drawback of the Information Agent was the fact that it visibly slowed down the performance of slower PC's (circa 300MHz) on the network. However, as specified previously, only one Information Agent needs to be active on a segment at any given time.

The Task Agents were able to abstract away from the physical details of frames because of the work carried out by the Information Agents. This allowed them to concentrate on achieving goals and solving problems where necessary. They were more in tune with the type of details a network manger would expect and because of their ability to communicate via KQML, they were able to obtain a complete view of the network and also solve problems collectively. They also required very little processing time in order to carry out their duties. The traffic generated by the Task

Agents was minimal and a further benefit was the fact that the traffic was distributed fairly evenly throughout the network, so that no one station or segment was put under undue pressure. Another major benefit of the Task Agents was their use of Rule Bases to detect and solve problems. This allowed for total flexibility on the part of the network manager. It was proven to be very easy to adjust rules and new rules could also be added without much difficulty. This helps reduce the amount of knowledge needed to successfully manage a network which in turn reduces the amount of expertise required by a network technician.

The greatest benefit of the agent-based network management system is in the intelligent pre-processing of data. Before any data is presented to the network manager, it is first put into a format which is of the most use to the manager. Each of the three agents in the agent architecture abstracts away from the original basic Ethernet frames, so that by the time the data is presented by the Interface Agent, it is in a useful format. The exact format is flexible as it can be controlled by entries in the Rule Base. A useful format may be as simple as a diagram showing daily traffic or it may mean that data is not presented at all to the network manager. For instance, when the agent-based management software detects corrupt frames being sent from a PC, the PC is rebooted. If this solves the problem then the network manager will not be informed as this is specified by an entry in the Rule Base. SNMPc on the other hand, will simply record and display all corrupt frames when requested. In an information age with virtually exponential growth in the World Wide Web one of the biggest challenges people face is managing the amount of information they are presented with, without sacrificing large amounts of time.

Another point worth noting is the simple fact that the better designed a network is, the less likely errors are to occur. As was previously mentioned, the test network used was under utilised. This avoided many of the problems (speed, segment over-utilisation), which many networks suffer from. However, there are obviously cost and time issues involved when setting up the "perfect network".

The ability of the agent-based management system to actually solve problems without human interaction is one that may be the greatest benefit to network management in the long run. Although it was only able to solve a limited number of problems with no user interaction, this has huge implications for network maintenance costs and network up-times. Software is nearly always going to be faster than human interaction and the distributed nature of agents', along with their ability to communicate, means that the potential is there to solve global problems. This further reduces the amount of information required by a network technician in order to successfully manage a computer network.

However, it should be noted that SNMP has been used and is a proven technology for larger, more heavily utilised LAN's. The same cannot be said for the agent-based management system. Also, because the software is placed on individual PC's, when an Information Agent is active on a PC, it is fighting for resources with other applications on the PC. It has been shown that the number of frames dropped by the Information Agent during normal PC use was very small. However, as previously stated the network was not being heavily utilised and was well segmented. Also, when the Information Agent was active, it did visibly affect the PC's performance. Obviously this was worse with slower PC's. There is also the possibility that (accidentally or on purpose) a user will stop an agent from running on a PC. This problem will potentially increase with the advent of Gigabit Ethernet, although one can safely assume that processor speeds will continue to increase. Leading on from that, although the test network used was well segmented with not more than 32 PC's on a segment, this will not always be the case. If the segment size is very large then faster processors will be needed to allow the Information Agent to monitor all frames. However, there is also a problem with the agent-based software if the segment size is too small. As has been stated previously for the best utilisation of the agent-based network management system, each segment should have one Information Agent active. Obviously the smaller the segment size, the more Information Agents are needed and this increases the total processing cost of the system.

6.1 Future Research

Although the agent based software shows much promise, it may be better placed on switches and routing devices. This would allow the agents complete access to MIB's, which already exist and have been proven to be useful, and it would also mean that there should not be the problem of dropped frames and PCs' being slowed down. A further benefit would be the direct interaction between the agents and the devices in order to detect and solve problems. It would also be worth researching the possibility of integrating the Information Agent more closely with the NIC's drivers on a PC. This would mean the agent would be closer to the kernel and should allow for faster processing of frames. Further research could also include researching whether it would be possible to have the agents communicate with network hardware providers. This would offer great benefits as if a problem was detected on a device, the device manufacturer might be able to provide assistance via its own agents. Obviously, performatives would need to be agreed, as would an agent communication language. A lot of research is currently being carried out into mobile agents for network fault diagnosis and this solution may well provide the long-term key.. Agents would only reside where they are needed and they could be specific to particular hardware, thus providing a huge range of expertise. This would also greatly suit the increase in the ability to control network devices remotely and the ever-increasing size and complexity of today's networks.

In the general computing sense it seems certain that software agents are here to stay and it is most likely that agent technology will have a positive impact on network management in the mid to long term.

References

[1] Kay A. Computer Software. Scientific American vol. 3, pp 53 - 59, 1984

[2] Franklin S. & Graesser A. Is it an agent or just a program?: A taxonomy for Autonomous Agents, 1996

[3] Sycara et al. Distributed Intelligent Agents. IEEE Expert, 1996

[4] Wooldridge M. Agents and Software Engineering. In *AI*IA Notizie* Xi vol 3, pp 31 - 37, 1998

[5] Nwana A. S. Software Agents: An Overview. Knowledge Engineering Review, pp 205-244, 1996

[6] Weiss G. Multiagent Systems: A modern approach to Distributed Artificial Intelligence, 1999

[7] Russell S. & Subramanian D. Provably bounded-optimal agents. Journal of AI Research, vol. 2 pp. 575-609, 1995

[8] Brooks A.R. Intelligence without representation, Computers and Thought, IJCAI, 1991

[9] Ferguson I.A. TouringMachines: Autonomous Agents with Attitudes. IEEE Computer, vol. 5 pp 25, 1992

[10] Wooldridge M. & Jennings N. R. Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 1994

[11] Rao A.S. & Georgeff M.P. BDI Agents: from theory to practice, Proc. 1st International Conference on Multi-Agent Systems, San Francisco, California, pp. 213-319, 1995

[12] Wooldridge M. & Jennings N.R. Pitfalls of Agent-Oriented Development. Agents '98: Proceedings of the Second International Conference on Autonomous Agents, ACM Press, 1998

[13] Nwana H.S. & Ndumu D.T. A Perspective on Software Agents Research, 1999

[14] Etzioni O. Moving up the information food chain: Deploying softbots on the World Wide Web. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1996

[15] Jennings N.R. Agent-Based Computing: Promise and Perils. Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99), Stockholm, Sweden, 1999

[16] Labrou Y. et al. Agent Communication Languages: The Current Landscape. IEEE March/April, 1999

[17] Finan T. et al. Draft Specification of the KQML Agent-Communication Language. DARPA Knowledge Initiative External Interfaces Working Group, 1993

[18] Case J. D., Fedor M., Schoffstall M. L. & Davin C. Simple Network Management Protocol. RFC 1157, 1990

[19] OSI. ISO 9595 Information Technology, Open System Interconnection, Common Management Information Protocol Specification, 1991

[20] Yemini Y. The OSI Network Management Model. IEEE Communications Magazine, pp 20 - 29, 1993

[21] Held G. Ethernet Networks: From 10 Base-T to Gigabit. Wiley publishing, 1998

[22] Baldi M., Silvano G. & Picco G.P. Exploiting Code Mobility in Decentralized and Flexible Network Management, First International Workshop on Mobile Agents 97 (MA '97), Berlin, Germany, Apr. 1997

[23] Davison R.G., Hardwicke J.J., & Cox M.D.J. Applying the agent paradigm to network management. BT Technology Journal, Vol. 16 No. 3, 1998

[24] Guha R.V. & Lenat, D.B. Enabling Agents to Work Together. Communications of the ACM, 37(7):127 - 142, 1994

[25] Genesereth, M.R. & Nilsson. N. Logical Foundations of Artificial Intelligence pp 325 - 327. Morgan Kaufmann Publishers, 1987

[26] Fikes, R.E. & Nilsson, N. STRIPS: A new approach to the application of theorem proving and problem solving. Artificial Intelligence, 5(2):189 - 208, 1971

[27] Vere, S. & Bickmore, T. A basic agent. Computational Intelligence, vol 6 pp 41 – 60, 1990

[28] Agre, P. & Chapman, D. PENGI: An implementation of a theory of activity. In Proceedings of the sixth national conference on Artificial Intelligence, pp 268 - 272, 1987

[29] J.E. White. Mobile Agents. Software Agents. MIT Press, 1996

[30] R.S. Grey. Agent Tcl: A transportable agent system. In Proceedings of the CIKM '95 workshop on Intelligent Information Agents, 1995

[31] Nordine M. & Unruth A. Facilitating Open Communications in Agent Systems: The Infosleuth Infrastructure. Proceeding from the fourth International Workshop on Agent Theories, Architectures and Languages, 1997

[32] Bradshaw J.M. Kaos: Toward and Industrial-Strength Open Agent Architecture. Software Agents, 1997

[33] Leake D.B. Combining Rules and Cases to Learn Case Adaption. Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, 1995

[34] Aamodt A. & Plaza E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications, Vol. 7, No. 1, 1995

[35] Peterson L.L & Davie B.S. Computer Networks: A Systems Approach. pp 121 – 127, Morgan Kaufmann, 1996

[36] Picard R.W. & Healye J. Affective wearables. Personal Technologies, 1(4):231 – 240, 1997

[37] Maes P. Agents the Reduce Work and Information Overload. Communications of the ACM, 37(7), pp 31 – 40, 1994

Appendices

Appendix A. Some Rules Used in Agent-based Network Management System

1)
```
If
        Average segment usage > 30%
Then
        Inform Network Manager
```

2)
```
If
        Inform network manager
Then
        Display on management station
```

3)
```
If
        Urgent inform network manager
Then
        E-mail network manager AND
        Display on management station
```

4)
```
If
        Network segment down
Then
        Urgent inform network manager
```

5)

```
If
        Number of corrupt frames > 10 AND
        Duration > 5
Then
        Corrupt Application
```

6)

```
If
        Corrupt Application
Then
        Reboot PC
```

7)

```
If
        Duration passed >= 168 hours
Then
        Check ability
```

8)

```
If
        User logged on
Then
        Ability = ability – 5
```

```
9)  If
            Single PC segment down >= 168 hours
    Then
            Inform network manager
```

```
10) If
            Standard day
    Then
            Record time between 09:00 AND 21:30
```

```
11) If
            Segment = 1 OR segment = 3 OR segment = 4
    Then
            Consider similar
```

```
12) If
            Segment = 2 OR segment = 6 Then
    Then
            Consider similar
```

```
13) If
            Segment = 5 OR segment = 7 OR segment = 8 OR segment = 9
    Then
            Consider similar
```

```
14)  If
         Similar segments with different utilisation
     Then
         Determine PC's causing problem AND
         Inform network manager
```

```
15)  If
         Time elapsed >= 168 hours
     Then
         Sustained period
```

```
16)  If
         Utilisation > 30%
     Then
         Heavy use
```

```
17)  If
         Traffic difference > 15% AND
         Consider similar AND
         Sustained period
     Then
         Similar segments with different utilisation
```

18) If

NOT Single PC segment AND

No Task Agent communicating

Then

Segment down

19) If

NOT Single PC segment AND

Main Task Agent communicating AND

Task Agent not communicating

Then

Node down

20) If

Node down

Then

Record

Appendix B. Explanation Facility for Rules Used

1) The average segment usage is greater than 30 percent of the segment bandwidth.

2) When informing the network manager, the appropriate message will be displayed on the network management station where the interface agent resides.

3) When the network manager needs to be informed of an urgent message, the message will be displayed by the interface agent and also e-mailed to the network manager.

4) Because a segment of more than one PC has gone down, the network manager is being informed urgently.

5) During a period of 5 minutes, more than 10 corrupt frames were detected from a single PC. An application is corrupt.

6) Because a corrupt application has been detected on a PC, the PC has been rebooted.

7) A PC is rechecking its ability to monitor its segment as at least a week has passed since it last performed the check.

8) A user is logged onto a PC so 5 has been subtracted from its monitoring ability.

9) The network manager is being informed, as a PC on the network has not responded for a week or more.

10) On a standard day, the network will be monitored between 09:00 and 21:30 as no users will be on the network outside these periods.

11) Because segments 1, 3 and 4 have the same number of PC's and are used for the same type of applications, they are being considered as similar and should not show too much of a difference in utilisation.

12) Because segments 2 and 6 have the same number of PC's and are used for the same type of applications, they are being considered as similar and should not show too much of a difference in utilisation.

13) Because segments 5, 7, 8 and 9 have the same number of PC's and are used for the same type of applications, they are being considered as similar and should not show too much of a difference in utilisation.

14) Because similar segments have displayed different utilisation the system is suggesting that some PC's be moved from the heavily utilised segment to a less utilised segment(s).

15) A sustained period is considered to be one week or greater.

16) Heavy use is considered to be utilisation greater than 30%.

17) Segment changes are suggested as a number of segments that are considered similar have a traffic difference of greater than 15% over a sustained period.

18) Because no task agents have communicated with the interface and there is more than one task agent on the segment, the segment must have a problem.

19) A node has gone down on a segment. Any node that goes down will be recorded but the network manager will not be informed by default as the PC may simply have been switched off.

20) If a node has gone down, then record the details.

Appendix C. Some Inter-Agent KQML Messages

1) (advertise : sender A :receiver B :language VB

:ontology NetMan :content "Ability (24.3)")

Sender A is advertising its new ability of 24.3

2) (ask-if :sender A :receiver B :language VB

:ontology NetMan :content "Segment Load" :reply-with "load")

Agent A is asking agent B what the load on B's segment is. B is expected to reply with "load".

3) (tell :sender B :receiver A :language VB

:ontology NetMan :content "28" :in-reply-to "load")

Agent B tells Agent A that its segment is under 28% load.

4) (tell :sender A :receiver B :language VB

:ontology NetMan :content "end")

Agent A wants agent B to stop monitoring the network segment

5) (end :sender A :receiver B :language VB

:ontology NetMan :content "better(32.4)")

This was used to replace message 4. Agent A tells Agent B to stop monitoring the network as Agent A has a better ability of 32.4

6) (rule :sender A :receiver B :language VB

:ontology NetMan :content "change(16,25)" :reply-with "changed")

Agent A tells Agent B that the condition for firing rule 16 should be changed to 25. Rule 16 determines what is defined as heavy use on a segment.

7) (tell :sender B :receiver A :language VB

:ontology NetMan :content "25" :in-reply-to "changed")

Agent B replies to Agent A

8) (rule :sender A :receiver B :language VB

:ontology NetMan :content "change(3,AND notify managers pager)" :reply-with "changed")

Agent A tells Agent B that rule 3 should be changed so that in the event of an urgent need to inform the network manager, he should also be notified by pager.

9) (rule :sender A :receiver B :language VB

:ontology NetMan :content "change(12,OR segment = 14)" :reply-with "changed")

Agent A tells Agent B that another segment has been added to the group.

10) (urgent :sender A :receiver B :language VB

:ontology NetMan :content "All Task Agents down" :reply-with "urgent")

Agent A sends an urgent message to agent B.

11) (tell :sender B :receiver A :language VB

:ontology NetMan :content "received" :in-reply-to "urgent")

Agent B replies to A's message.

12) (tell :sender A :receiver B :language VB

:ontology NetMan :content "alive" :reply-with "alive")

Agent A is sending an alive message to agent B

13) (tell :sender B :receiver A :language VB

:ontology NetMan :content "alive" :in-reply-to "alive")

Agent B sends an alive message of its own

14) (error :sender A :receiver B :language VB

:ontology NetMan :content "00:8c:2f:3a:aa:9c, 00:4a:0c:fc:00:2a, length")

Agent A tells Agent B that a frame has been received with an invalid length field. The source and destination MAC addresses are also included.

15) (error :sender A :receiver B :language VB

:ontology NetMan :content "00:8c:2f:3a:aa:9c, 00:4a:0c:fc:00:2a, jabber")

Agent A tells Agent B that a frame has been received that is too long. The source and destination MAC addresses are also included.

16) (find :sender A :receiver B :language VB

:ontology NetMan :content "00:8c:2f:3a:aa:9c")

Agent A queries if Agent B knows the IP address of the enclosed MAC address.

17) (found :sender B :receiver A :language VB

:ontology NetMan :content "193.222.222.222")

Agent B replies with the IP address.