Information and Knowledge Management
ISSN 2224-5758 (Paper) ISSN 2224-896X (Online)
Vol.5, No.3, 2015

Development of a Proactive Fault Diagnosis for Critical System

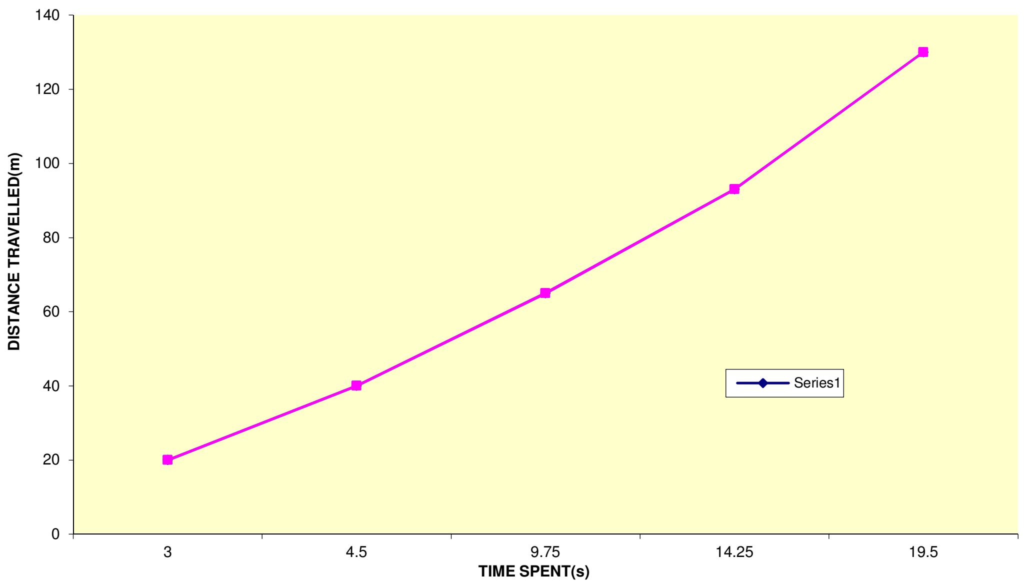
Mr Alabi S .A[1] DR Adeosun O. O[2] Oloyede T.David[3]
1.Department of Computer Science,Osun State College of Technology, Esa-Oke, Nigeria
2.Department of Computer Science and Engineering,Ladoke Akintola University of Technology, Ogbomoso,Nigeria
3. National Open University of Nigeria,Osogbo Study Center

ABSTRACT
Large-scale network environments, such as the Internet, are characterized by the presence of various devices connected at various remote locations. There is a scenario of main office connected to different branch offices in another town and cities, with the presence of central administrative system at the main office. Any problem at branches is reported to the main office, due to availability of enough resources there. However, few support tools have been developed to allow the administrators at the central office to remotely control and monitor the computers at the branches.

Even, in local area network environment, diagnosing the computers on the network is always a big problem for the administrator, as he/she moves from one computer to another, running the diagnostic program and collecting report for each machine tested. This is strenuous and time consuming.

To help address these problems, I have employed the concept of mobile agent to design an architecture that can remotely perform various checks and tests on computers on network, and report its findings to the server administrator as central location. This architecture was implemented with Java, using Jini lookup service to establish communication between the computers. The agent tasks were implemented in C programming language.

The result of this research work shows that the use of mobile agent for remote maintenance of computers on network was found to provide an improved, efficient, and dynamic diagnostic management system. All the same, it has proven to be a substantive contributor to efficient network management.



GRAPH OF DISTANCE TRAVELLED AGAINST TIME SPENT BY MOBILE AGENT

INTRODUCTION

Diagnostic programs serve the purpose of detecting or locating faults in the logic circuitry. The use of these troubleshooting software's is one of the techniques used to effect microcomputers systems maintenance, and to introduce software redundancy needed for building fault-free computers. A typical diagnostic program is expected to monitor the status of computer system including the peripheral devices connected to it, and endeavor to give the report of the status of devices tested. This allows the users to determine the status of his computer and know the next line of action if not satisfied with report condition. Usually, a diagnostic program is applicable to diagnose a transient fault, i.e. temporary malfunctioning of a computer system.

This has created a need for new ways of structuring application to provide cost-effective and scalable models. Most existing implementation for fault diagnosis on a computer network use the traditional client- server (CS) design. But due to some limitation in client server, an autonomous mechanism is developed for conducting the fault diagnosis ad report known as mobile agent.

Mobile agent are autonomous software that migrates from one host to another to perform a specific task on behalf of an administrator. Fault diagnosis involve the use of mobile agent to carry out troubleshooting and report any found error to an administrator over the internet. Many approaches have been developed to support fault diagnostic system. Among which are client server and mobile agent. While client server paradigm was the approach used in the fault diagnosis for the past two decades, accuracy, reliability, security and effective performance have not been totally met by the client sever paradigm, hence, the need for mobile agent. On the other hand, despite the growing effort been invested on the development of mobile agent systems, these system have not proved to be reliable. Problem such as host failure, communication failure and loss of agents and their states exist as they do in other distribution system (Chan 2000) security and reliability issue.

PROJECT MOTIVATION

The business world is nearing the culmination of a trend away from stand-alone computers towards an entirely connected world where data produced or maintained in one location is readily available to everyone in the enterprise who needs data for day-to-day management or decision-making. Today, nearly all-organizational computers are attached to such form of network. However, due to proliferation of computers, network acquisition and integration, and management of these clients have become time-intensive and cost-prohibitive. As part of these management functions are clients' device maintenance and troubleshooting operations, which this project focused on.

According to Aderounmu (2002), determining the status of these Personal Computers (PCs) in a network environment is always a big problem as the network administrator needs to move from one machine to another, running the diagnostic software and collecting reports for decision-makings. Time and efforts are spent to do this.

Client/Server technology solutions are proffered to alleviate this dilemma (Case et al, 1990). The conventional diagnostic software's (such as Norton Utilities) are installed on the server, and the clients send request to the server whenever diagnosis and maintenance tasks are to be performed. The server in turn sends a response to the request and the "handshaking" occurs again and again. Each request/response of the traditional diagnostic programs requires a complete round trip across the network.

There are problems with this model, which the proposed designed architecture addresses:

Bandwidth Utilization: By moving a piece of code from one host to another, via the server, the repetitive request/response handshake is eliminated.

Unrealistic Network Connection: The architecture successfully handles the problem created by intermittent or unreliable network connections. The agents are built to work 'off-line' and communicate their results back whenever a system is 'on-line'.

Reduced Communication Costs: Some diagnostic programs contain large disk space on server, with varieties of diagnostic utilities. Transferring the whole program to a host from the server (through handshake) can be very time-consuming and become a bottle neck in the network. Imagine having to transmit Norton Utilities 2.0. With different functional parts, such as System Doctor for VirusScan, Disk Doctor to repair Disk and Registry tracker to watch over the registry, from the server to client, just to use only registry checker. It is much more natural and economical to get a developed agent to 'go' to host, do a local search of its needs and perform relevant diagnostic task. This provides a much cheaper alternate in terms of network bandwidth and time.

Another motivation for this project is the unavailability of enough local resources. The processing power and storage in the local machine may be very limited thereby necessitating the use of mobile agents. The agent architecture supports asynchronous computing (Todd, 1998). The agents are set off from a host to all other clients/hosts and the results of its diagnosis may be reported back at later time. They can operate when a host is not even connected.

THE PROJECT OBJECTIVES

The major objectives of this project work design are to:

Provide a flexible distributed computing architecture.

Design an architecture that remotely carries out fault diagnosis of microcomputers interconnected on network in a dynamic and more flexible manner.

Develop an intelligent software agent that could transverse over the network, monitor the status of each computer on the network and give quick alarm in case of malfunction.

Remove the delay associated with collection of diagnostic reports of computers on network, thus enhance quick decision-making process.

Improve the efficiency of networking. Since the mobile agent can move across the network solely to the desired locations, thus reducing network traffic overload. This paves way for efficient utilization of available bandwidth.
Reduce the risks associated with technical jobs. In a wide area network, where branches are connected to main office, breakdown of computer(s) in a branch can simply be handled by sending down the diagnostic mobile agent from the
main to the branch office, which will locally test the computer and if possible detect the faults. Thus reduce the network technician in the main office travel risks.
To establish a network that is fault tolerant. In a client/server relationship, if a network or server fails during a request, it's difficult for the client to retain the situation and re-synchronize with the server because the network connection would have been lost. In this project design, the mobile agent software does not need to maintain permanent connection and its state is centralized within itself. Therefore failures are generally easier to deal with.

PROJECT METHODOLOGY

The Mobile Agent Diagnostic software framework consists of two components:

- System Diagnosis Component
- Mobility Infrastructure Component

The system diagnosis component determines the status of available hardware resources, that is testing, monitors and knows when things go wrong with any of the peripherals. The mobility infrastructure is responsible for sending and receiving the mobile agents, with two migration techniques:

- transportation of the program code with its state to the network.
- activation of the code on local machine.

The system diagnosis component is built with C compiler, so as to take advantage of its ability to access hardware resources. The expected hardware resources to monitor or test include keyboard, mouse, printer, floppy drive, hard drive, visual display unit, system clock, communication ports (parallel and serial) and memory. In addition, utilities services such as getting equipment details, CMOS setup details, write data to CMOS RAM, test interrupt activation and handle Shift keys status, are provided.

The mobility infrastructure component of the framework is implemented in JAVA, using JINI (which is based on Remote Method Invocation), Object Serialization and JavaBeans API component model. Java, being an Internet application language, facilitates a TCP/IP socket-based connection between the source and the destination during the agent mobility.

SCOPE OF THE PROJECT

The diagnostic mobile agent software is architecturally designed to handle only remote maintenance aspect of network fault management. It could perform tests remotely on computers on network, detects any malfunction, gives report of tests and monitors the status of interconnected microcomputers and their peripheral devices. It tries to repair any detected fault, which if fails, direct the repair to technicians. Virus detection utilities are not included.

The diagnostic mobile agent software was implemented on local area network consisting of minimum of 10 clients and 1 server in The University of Ibadan, Ibadan cyber café.

PROJECT LAYOUT

The remaining work is organized as follows: Chapter two discusses the literature review of client server and mobile agent platform as well as the choices of the programming language used. Chapter three discusses the analysis of mobile agent and client server. Features like speed, distance traveled and time taking was used for the analysis. Chapter four discusses system implantation and testing, software installation, the web test environment and what the system does and the comparison between the two platforms; while chapter five discusses summary, conclusion and future recommendation of this work.

LITERATURE REVIEW

DIAGNOSTIC SOFTWARES

The use of microcomputer troubleshooting software is one of the techniques used to effect microcomputer systems maintenance. This is an aspect of computing that involves some tests and measurements carried out to effect repairs or replacements; to keep the performance of the system. Ajulo, (2001) stressed that properly administered maintenance pays for itself many times over and reduces problem behavior, data loss, component failures and minimizes downtime periods.

According to Aderounmu et al. (2002), a diagnostic program usually works by performing a low-level probe on the status of the computer system and the peripheral devices attached to it. It then records all the status information of each device tested and sends the result of the diagnosis to compile a report for assessment. From the report, the user usually determines if he is satisfied with the status of all his devices or not, and hence take appropriate action.

Ajulo (2001) further describes Diagnostic software's as programs that are used to check the microcomputer components. Such software's are capable of identifying faulty or defective hardware. For example, if any of the modular cards are not making proper contact on microcomputer buses, the diagnostic software would simply flag error messages for recognition or technicians to take steps that correct such problems.

Computer peripherals and components require regular maintenance for proper use and effective functioning (Baldi, 1998). Diagnostic software of various types effectively and efficiently troubleshoots these components, gathering clues and applying deductive reasoning to isolate the problem by performing various tests on the underlying circuitry structure of the component. For example, if space for storage is becoming smaller, diagnostic software is used to compress data or double space Hard disk for temporary storage space enhancement. Bad sectors or defective heads and cylinders can be masked, packed or moved to improve on the Hard disk drive performance.

FEATURES OF CONVENTIONAL DIAGNOSTIC SOFTWARE

There are various types of diagnostic software's that are available today. Some are for disk management and repair (Disk Manager), monitoring computer's startup and shutdown processes, health status, registry management, file management, scan, detect and clean virus, and other tasks that contribute to proper functioning of computer system. Features of common ones are reviewed in this section.

PC HANDYMAN (PCH)

PC Handyman (PCH) by Symantec is a computer expert inside the computer that watches for problems or things that go bump-in-the-night and then lets the users know or automatically fixes them÷ It provides glossary of terms that helps the users to understand their system. It also helps to customize the desktop so that it looked and worked just like the owner wants it to. PCH monitors the computer's startup and shutdown process, checks hard drive for efficient operation and the quality of its surface to anticipate "bad sector" problems. It further checks for the "correctness" of. DLL and EXE files for errors or file references.

FIRST AID 97

First Aid 97 (FA 97) by Cyber media has many features that check for anomalies and provide fixes and solutions when available. Check-up will ensure that applications work properly, peripherals have no conflicts, multimedia works properly and computer settings are set for optimal performance.

FA97's Windows Guardian prevents crashes. It auto fixes shortcuts, auto refresh rate, optimizes the file system to run multimedia applications, animation (speed at which widows is minimized and maximized), menu popup rate, and other problems.

The limitation with First Aid 97 is that it would not install under Dos on the Windows 95/98 system, until Windows 95 drivers were in place.

NORTON UTILITIES

NU2, from Symantec, is designed to assist the user in any number of computer "housekeeping" duties from making the system run more efficiently to finding out what causes a problem and suggesting how to fix it. NU2 includes CrashGuard with Anti-freeze, which saves work running in case of a "system crash" or freeze. NU2 has the following components:

Norton System Disk: This will scan virus, detect disk and system problems and will take preventive measures to bail out the system when in serious trouble.

Norton Disk Doctor: This is used to repair a large number of computer problems, from logical disk structures to the hard drive's surface, and helps in determining when the hard drive is starting to go bad.

Norton Registry Tracker: This keeps a running file of changes that take place in the registry so that the user will always be informed of what is happening to his/her system. The registry file is vital and is the heart of operating system.

Norton Registry Editor: This allows editing of the registry, but also allows one to UNDO any changes that have been made where REGEDIT will not.

Norton File Compare: This shows the changes in a file, like .INI file after a program is added to a system.

Speed Disk: This works like Microsoft's Defrag, but also defrags the Swap File and "verify" its actions.

CHECKIT – DIAGNOSTIC KIT VERSION 4

The DOS version, Check It 4 by Touchstone is very useful since it boots and runs from the floppy drive. This gives the technicians and casual user a portable diagnostic tool. It is great for verifying the performance of a new or used computer that one is thinking of purchasing. The "kit" explains what tests can be performed and how to interpret the test results. The following features are embedded in Check It.

Burin-In Certification: An automated batch-testing program that tests a computer repeatedly to reveal potential problems. If a new system is built, it can identify problem on the motherboard within 24 to 48 hours that otherwise may not have surfaced for a month.

RAM exam: Is a multilevel diagnostic program that tests six levels of memory and identifies effective RAM chips.

Advisor: Used to troubleshoot installation of a new hardware card before taking of the system. There is a long list of adapter cards from various manufacturers included with the program.

Shopper: This informs the user of system requirements for different types of software's that are anticipated installing. The list in the program package can also be updated.

LIMITATIONS OF CONVENTIONAL DIAGNOSTIC PROGRAMS

According to Mennie (1998), conventional diagnostic programs are designed, installed and implemented as stand-alone applications, performing their functions on local computers. They could only solve problems locally and do not support Distributed Problem Solving (DPS), whereby several centralized applications, each capable of addressing a certain aspect of a problem, are tied together by a communication system. So, determining the status of each personal computer in a network requires the network administrator to move from one machine to another, running the diagnostic software. This is always a big problem as time and efforts are spent to do this. This limitation motivates the design of an architecture that uses mobile agents for remote maintenance.

COMPUTER COMPONENTS DIAGNOSTICS

To understand the working principle of any diagnostic program is important to examine various ways in which some computer components are designed and programmed by manufacturers. A program could only test and examine computer components through segment addresses (Michael, 2002). So, this section gives detail analysis of underlying structure and low-level address organizations and references required to control the system's hardware.

MEMORY DIAGNOSTIC

Semiconductor memory devices tend to fall into two main categories: "read/write" and "read-only". Read/Write memory is the one, which its contents can be modified at, will, while Read-only memory on the hand, can only be read from; an attempt to write on to such a memory will have no effect on its contents. Each location in semiconductor ROM and RAM has its own unique address (Michael, 2002). At each address, a byte is stored. Each ROM, RAM accounts for a particular block of memory, its size depending upon the capacity of the ROM or RAM in question. For example, 486-based system with 4Mbytes of RAM is arranged in four blocks of 1Mbyte. Each 1Mbte of RAM is provided with a single-in-line memory module (SIMM) fitted with nine DRAM chips. Each DRAM chip has a capacity of 1,048,576 bits. The organization of this memory, which occupies the address range from 000000 to 3FFFFF is shown in Table 2.1, where 'U1', 'U2', etc are the component designations for the individual RAM chips (total of 36 chips fitted) (Michael, 2002).

Table 2.1 4 X 1Mbute SIMM Memory Example (total 4Mbyes)

Block	Address Range	D7	D6	D5	D4	D3	D2	D1	D0	Parity Bit
0	000000-0FFFFF	U8	U7	U6	U5	U4	U3	U2	U1	U9
1	100000-1FFFFF	U17	U16	U15	U14	U13	U12	U11	U10	U18
2	200000-2FFFFF	U26	U25	U24	U23	U22	U21	U20	U19	U27
3	300000-3FFFFF	U35	U34	U33	U32	U31	U30	U29	U28	U36

BIOS ROM

The BIOS ROM that contains the low-level code required to control the system's hardware is programmed during chip manufacture by the BIOS originator, thus programming of the ROM is irreversible. Hence, the only way of upgrading the BIOS is to remove and discard the existing chips and replace them with new ones. The BIOS ROM invariably occupies the last 64Kbytes or 128Kbytes of memory (from F0000 to FFFFF or E0000 to FFFFF respectively). It is sometimes based on two chips; one for the data stored at the odd addresses and the other for the data store at the even addresses.

According to Michael (2002), at some point, it is necessary to upgrade the BIOS ROM on a machine for various reasons but most centers on the need to make software recognizes significant hardware upgrades for example to make use of IDE hard drive. Several manufacturers such as Compaq have produced ROM BIOS code for use in their own equipment. This code must, be compatible with IBM's BIOS code. (shown in Appendix A). Several other companies such American Megatrends – AMD, Award Software and Phoenix Software have developed generic versions of the BIOS code, which have been incorporated into numerous clones and compatibles.

RAMDOM ACCESS MEMORY (RAM)

The PC System board's read/write memory provides storage for the DOS and BIOS as well as transient user programs and data. In addition, it is used to store data, which is displayed on the screen. Semiconductor RAMs are divided into "static" and "dynamic" types. Static types are generally CMOS (Complimentary Metal Oxide Semiconductor) – for power consumption) and are based on bistable memory cells. Each cell will retain a stored bit (logic 1 or logic 0) for as along as the power is left connected.

The dynamic memories (DRAM) are generally NMOS types which utilize charge storage within a semiconductor junction. To prevent the stored charge leaking away (in which case the memory would lose its contents), the charge is periodically 'refreshed' by a continuous process of reading and writing. This process takes place automatically.

MEMORY TERMINOLOGY

Conventional RAM

The PC's RAM extending from 00000 to 9FFFF is referred to as a conventional or base memory. This 640K of read/write memory provides storage for user programs and data as well as regions reserved for DOS and BIOS use.

Upper Memory Area

The remaining within the 1Mbyte direct addressing space that is, that which extends from A0000 to FFFFF, is referred to as the 'upper memory area'
(UMA). The UMA itself is divided into various regions (depending upon machine's configuration) including that which provides storage used by video adapters.

VIDEO RAM

Video RAM is associated with the video.graphic adapter. The video RAM occupies the lower part of the upper memory area and its precise configuration depend upon the type of adapter fitted.

Extended Memory

Memory beyond the basic 1Mbyte direct addressing space is referred to as 'extended memory'. This memory can be accessed by a '286 or later CPU operating in 'protected mode'. In this mode, the CPU is able to generate 24-bit addresses (instead of the real mode's 20-bit addresses) by multiplying the segment register contents by 256 (instead of 16). This scheme allows CPU to access addresses ranging from protected mode; a program can only access the designated region of memory. A processor exception will occur if an attempt is made to write to a region of memory that is outside of the currently allocated block.

Expanded Memory

Expanded memory was originally developed for machines based on the 8088 CPU, which could not take advantage of the protected mode provided by the 486 and later processors. Expanded memory is accessed through a 64K 'page frame' within the upper memory area. This page frame acts as a window into a much larger area of memory.

CMOS RAM

The PC-AT and later machine's CMOS memory is 64 bytes of battery-backed memory contained within real-time chip (a Motorola MC146818). According to Micheal, 2002, Sixteen of this memory area is used to retain the real-time clock settings while the remainder contains important information on the configuration of the system. The organization of the CMOS memory is shown in Table 2.2 below.

Table 2.2 CMOS Memory Organization

Offset (Hex)	Contents
00	Seconds
01	Seconds alarm
02	Minutes
03	Minutes alarm
04	Hours
05	Hours alarm
06	Day of the week
07	Day of the month
08	Month
09	Year
0A	Status register A
OB	Status register B
OC	Status register C
OD	Status register D
OE	Diagnostic status byte
OF	Shutdown status byte
10	Floppy disk type (drives A and B)
11	Reserved
12	Fixed disk type (drives o and 1)
13	Reserved
14	Equipment byte
15	Base memory (low byte)
16	Base memory (high byte)
17	Extended memory (low byte)
18	Extended memory (high byte)
19	Hard disk 0 extended type
1A	Hard disk 1 extended type
1B-2D	Reserved
2E-2F	Checksum for bytes 10 to 1F
30	Actual Extended memory (low byte)
	Actual extended memory (high byte)
32	Date century byte (in BCD format)
33-3F	Reserved

DOWN MEMORY LANE

The allocation of memory space within a PC can be usually illustrated by means of a memory map. A 8086 microprocessor can address any one of 1,048,576 different memory locations with its 20 address lines. It thus has a memory, which ranges from 00000 to FFFFF. Figure 2.1 below shows a representative memory map of a PC in which the memory is allocated as shown in Table 2.3.

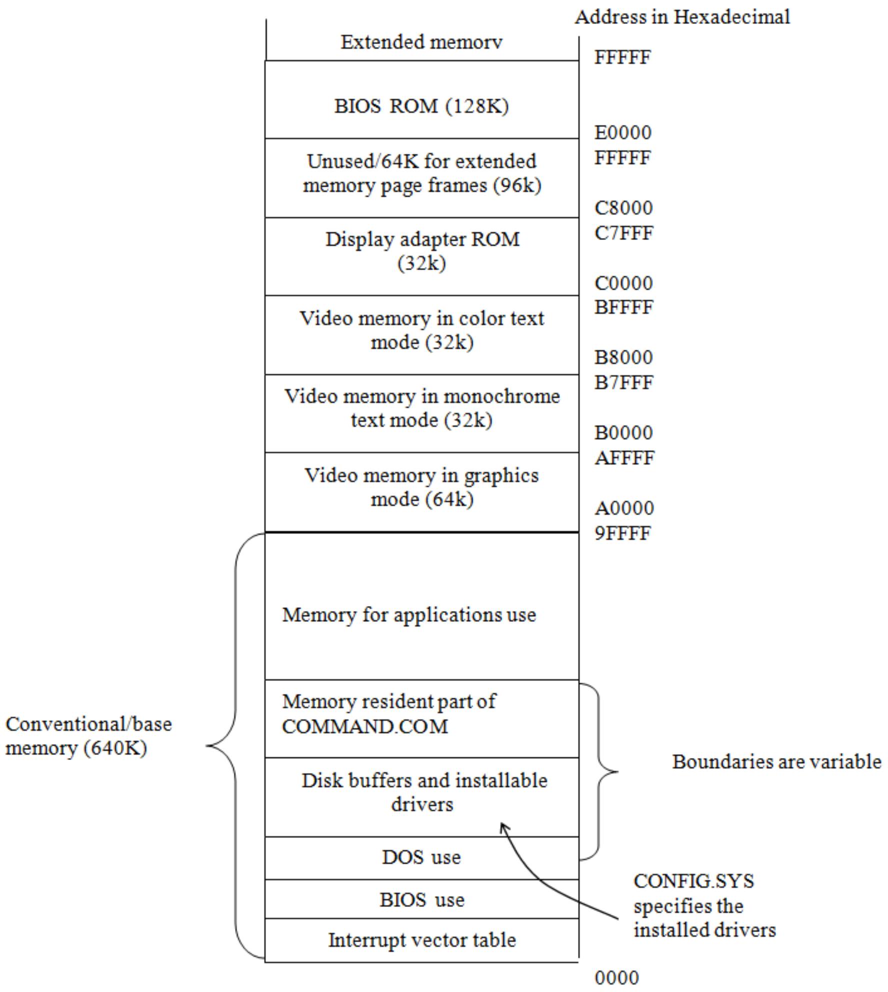

Address in Hexadecimal
Extended memory
FFFFF
BIOS ROM (128K)
E0000
FFFFF
Unused/64K for extended memory page frames (96k)
C8000
C7FFF
Display adapter ROM (32k)
C0000
BFFFF
Video memory in color text mode (32k)
B8000
B7FFF
Video memory in monochrome text mode (32k)
B0000
AFFFF
Video memory in graphics mode (64k)
A0000
9FFFF
Memory for applications use
Memory resident part of COMMAND.COM
Disk buffers and installable drivers
DOS use
BIOS use
Interrupt vector table
0000
Conventional/base memory (640K)
Boundaries are variable
CONFIG.SYS specifies the installed drivers

Figure 2.1: Representative Memory Map for a PC

Table 2.3 Memory allocation in a basic specification PC

Address range (hex)	Size (bytes)	Use
00000-9FFFF	640K	Conventional (base) memory
A0000-AFFFF	64K	Video memory (graphics mode)
B0000-B7FFF	32K	Video memory (monochrome text mode)
B8000-BFFFF	32K	Video memory (colour text mode)
C0000-C7FFF	32K	Display adapter ROM (EGA, VGA, etc)
C8000-DFFFF	96K	Unused (page frame for extended memory etc)
E0000-FFFFF	128K	BIOS ROM

USING RAM AND ROM TO FIND OUT ABOUT A SYSTEM

A number of memory locations can be useful in determining the current state of a PC or PC-compatible microcomputer. The programs and drivers that are present in the memory at any time can be known for example by using DOS MEM. The size of the expanded memory and the segment address of its page frame can also be viewed.

USEFUL RAM LOCATIONS

The contents of these memory locations are shown in Table 2.4 below

Table 2.4: Useful RAM locations

Address (Hex)	Number of bytes	Function
0410	2	Installed equipment list
0413	2	Usable base memory
0417	2	Keyboard status
043E	1	Disk calibration
043F	1	Disk drive motor status
0440	1	Drive motor count
0441	2	Disk status
0442	2	Disk controller status
0449	1	Current video mode
044A	2	Current screen column width
046C	4	Master clock count (incremented by 1 on each clock 'tick'
0472	2	Set to 1234 hex. During a keyboard re-boot (this requires <CTRL-ALT-DEL> keys
0500	1	Screen print byte (00 indicates normal ready status, 01 indicates that a screen print is in operation, FF indicates that an error has occurred during the screen print operation)

WHAT EQUIPMENT IS INSTALLED?

The machine's Installed Equipment List (Table 2.5) usually tells what hardware devices are currently installed in a system. The equipment list is held in the word (two bytes) starting address 0410. Figure 2.2 and Table 2.6 show how to decipher the equipment word.

Table 2.5 Equipment List Word at address 0410

Bit Number	Meaning		
0	Set if disk drives are present		
1	Unused		
2 and 3	System Board RAM size:		
	Bit 3	Bit 2	RAM size
	0	0	16K
	0	1	32K
	1	1	64K/256K
4 and 5	Initial video mode:		
	Bit 5	Bit 4	Mode
	0	1	40 column colour
	1	0	80 column colour
	1	1	80 column monochrome
6 and 7	Number of disk drives plus 1:		
	Bit 7	Bit 6	Number of drives
	0	0	1
	0	1	2
	1	0	3
	1	1	4
8	Reset if DMA chip installed (standard)		
9 to 11	Number of serial ports installed		
12	Set if an IBM Games Adapter is installed		
13	Set if a serial printer is installed		
14 and 15	Number of printers installed		

Address:	(hex)	0411		0410	
Contents:	(hex)	4	2	2	D
	(binary)	0100	0010	0010	1101
Bit Position:		15	8	8	0

Figure 2.2: Deciphering the equipment list

Table 2.6: Example
Table 2.6: Sample equipment list

Bit position	Status	Comment
0	1=set	Disk drives are present
1	1=set	This bit is not used
2	0=reset	These bits have no meaning
3	0=reset	Systems having greater than 256K RAM
4	0=reset	Initial video mode is 80 column
5	1=set	Color
6	1=set	Two disk drives installed
7	0=reset	
8	0=reset	DMA controller fitted (standard)
9	0=reset	Two serial ports installed
10	1=set	
11	0=reset	
12	0=reset	No IBM Games Adapter installed
13	0=reset	No serial printer installed
14	1=set	One printer attached
15	0=reset	

Interpretation:
Bit position 5 that is set to 1 indicates 80 column color video mode. Bit position 6 set to 1 indicates that number of drives is 2. The following code segment could be used to get installed equipment list from RAM and determine the number of floppy drives present:

```
DEF SEG = &H0
Equlo% = Peek(&H410)
Ndrive% = 1 +((equlo% AND 192)/64)
```

HOW MUCH BASE MEMORY IS AVAILABLE?
The amount of usable base memory can be determined from the two bytes starting address 0413. The extent of memory is found by simply adding the binary weighted values of each set bit position. Table 2.7 shows how this work.

Table 2.7: Determining the base memory

Bit position	Status	Value
9	Set	512K
8	Reset	256K
7	Set	128K
6	Reset	64K
5	Reset	32K
4	Reset	16K
3	Reset	8K
2	Reset	4K
1	Reset	2K
0	Reset	1K

WHAT ROM IS FITTED
The BIOS ROM release date and machine ID can be found by examining the area of read-only memory extending between the absolute locations FFFF5 and FFFFC. The ROM release information (not found in all compatible) is presented in American date format using ASCII characters.

WHAT KIND OF MACHINE IS IT?
The type of machine (whether PC-XT, AT, etc) is encoded in the form of an identification (ID) byte which is stored at address FFFFE. Table 2.8 gives the ID bytes for each member of the PC family (non-IBM machines may have ID bytes which differ from those listed).

Table 2.8: ID bytes for various IBM machines

ID Byte (Hex)	Machine
F8	PS/2 Models 35, 40, 65, 70, 80 and 90 (‘386 and ‘486 CPU)
F9	PC Convertible
FA	PS/2 Models 25, 30 (‘8086 CPU)
FB	PC-XT (revised versions, post 1986)
FC	AT, PS/2 Models 50 and 60 (‘286 CPU)
FE	XT and Portable PC
FF	Original PC

ROM AND RAM DIAGNOSTICS

The PC’s BIOS ROM incorporates some basic diagnostic software which checks the BIOS ROM and DRAM during the initialization process. The ROM diagnostic is based upon a known ‘checksum’ for the device. Each byte of ROM is successively read and a checksum is generated. This checksum is them compared with a stored checksum or is adjusted by padding the ROM with bytes which make the checksum amount to zero (neglecting overflow). If any difference is detected, an error message is produced and the bootstrap routine is aborted.

In the case of RAM diagnostics, the technique is quite different and usually involves writing and reading each byte of RAM in turn. Various algorithms have been developed which make this process more reliable for example ‘walking ones’ (Michael, 2002). Where a particular bit is ‘stuck’ that is refuses to be changed, the bootstrap routine is aborted and an error code is displayed. This error code will normally allow a diagnostic program to identify the particular device that has failed.

The Power-On-Self-Test (POST) code within the BIOS ROM checks the system (including system board ROM and RAM) during initialization. The POST reports any errors detected using a numeric code full list is given in Appendix A.

PARALLEL PRINTER INTERFACE DIAGNOSTICS

The PC’s parallel ports (LPT1 and LPT2) provide a very simple and effective interface, which can be used to link a PC to a wide range of printers and other devices such as external tapes and disk drives. This section explains the principles of parallel I/O and its basic faults finding.

Parallel I/O is used to transfer bytes of data at a time between a microcomputer and a peripheral device (such as printer). This method of I/O requires a minimum of hardware for example a single 8255 parallel I/O device and it is thus relatively easy and inexpensive to implement. The 8255 (or equivalent) is used to interface the PC’s system data bus to the external 8-bit data lines that link the computer to the printer. In addition, several other control signals are present in order to achieve “handshaking”; the process, which controls the exchange of data between the computer system and the printer. A basic handshake sequence is as follows:

The PC indicates that it is ready to output data to the printer by asserting the STROBE line.
The PC then waits for the printer to respond by asserting the ACK (acknowledgement) line.
When ACK is received by the PC, it places the outgoing data on the eight data lines.
The cycle is then repeated until the printer’s internal buffer is full of data.

The buffer may have to be filled several times during the printing of a large document. Each time, the port will output data at a fast rate but the printer will take an appreciable amount of time to print each character and thus will operate at a very much slower rate. The standard employs parallel data transmission (a byte is transferred at a time). The code segment below is used to read the status of PC’s parallel printer ports.

```
DEF SEG= &H40
Do
        Status%=PEEK(9) * 256 + PEEK(8) +1
        stat% =inp(Status%)
        If stat% = &H57 then Print “Printer Not Ready”
        If stat% = &H77 then Print “Printer out of paper”
        If stat%=&HF7 then print “Printer off-line”
Loop until stat%= &HDF
Print “Printer Ready”
DEF SEG
```

TROUBLESHOOTING THE PARALLEL PRINTER INTERFACE

Fault finding in a PC printer interface is usually quite straightforward and generally involves checking first that the printer is operating correctly (by using the printer ‘self-test’), and that no error condition exists for example paper out. It will usually be a fairly easy matter to decide which part of the interface (parallel port) is at fault. Where text is printed but characters appear to be translated resulting in gabled output, one or more of the data line signals may be missing. In this case, it is worth checking individual signal lines (D1 to D8) at each end of the cable. Special codes can be sent to a printer in order to determine the type style and page format. These codes can take the form of single byte characters (ASCII characters in the range 0 to 1f hex), or sequence of characters preceded by the ASCII character (27 decimal or 1B hex). The Table 2.9 below shows codes to control printer styles on Epson-compatible printers.

Table 2.9: Control Print Styles Characters

Characters	Effect
CHR$(15)	Condensed mode
CHR$(27) + "G"	Double strike mode
CHR$(27)+"E"	Emphasized mode
CHR$(27)+"4"	Italic mode
CHR$(27)+"S"+"0"	Subscript mode
CHR$(27)+"S"+"1"	Superscript mode
CHR$(18)	Cancel condensed mode
CHR$(27)+"F"	Cancel emphasized mode
CHR$(27)+"H"	Cancel double strike mode
CHR$(27)+"5"	Cancel italic mode
CHR$(27)+"T"	Cancel subscript/superscript mode

SERIAL COMMUNICATION PORTS DIAGNOSTICS

The PC's serial communication ports (COM1, COM2, etc) provide a means of linking the PC with the rest of the world. Data can be exchanged with remote host computers, 'bulleting boards' and a vast number of other PC users world-wide. Serial I/O involves transmitting a stream of bits, one after another from a PC to a peripheral device and vice versa. Serial input data must be converted to the system bus. Conversely, the parallel data on the bus must be converted into serial data before it can be output from the port. In the first case, conversion can be performed with a serial-input parallel-output (SIPO) shift register while in the second case, a parallel-input serial-output (PISO) shift register is required. Serial data may be transferred in either 'synchronous' or 'asynchronous' mode. In the former case, transfers are carried out in accordance with a common clock signal. Asynchronous operation, on the hand involves transmission of data in small "packets"; each packet containing the necessary information required to decode the data, which it contains. The most notable input and output signals used with serial I/O devices are shown in Table 2.10 below.

Table 2.10: Signals produced by a serial I/O device

Signal	Function
D0 to D7	Data input/output lines connected directly to the system data bus
RXD (or RD)	Received (incoming) serial data
TXD (or TD)	Transmitted (outgoing) serial data
CTS	Clear to send. This signal is taken low by the peripheral when it is ready to accept data from the computer system
RTS	Request to send. This signal is taken low by the UART (Universal asynchronous receiver/transmitter) when it wishes to send data to the peripheral.

PC DISPLAY DIAGNOSTICS

The video capability of a PC will depend not only upon the display used but also upon the type of "graphics adapter" fitted. The various display standards are shown in Table 2.11 below.

Table 2.11: Display Adapter Summary

Display Standard	Text capability (columns X lines)	Graphics capability (horizontal X vertical pixels)
MDA	80 x 25 monochrome	None
HGA	80 x 25 monochrome	720 x 320 monochrome
CGA	80 x 25 in 16 colours	320 x 200 in two sets of 4 colours
EGA	80 x 40 in 16 colours	640 x 350 in 16 colours
MCGA	80 x 30 in 16 colours	640 x 480 in 2 colours, 320 x 200 in 256 colours
8514/A (PS/2)	80 x 60 in 16 colours	1024 x 768 in 256 colours
VGA	80 x 50 in 16 colours	640 x 480 in 16 colours, 320 x 200 in 256 colours
SVGA	132 x 60 in 16 colours	1024 x 768 in 256 colours, 640 x 480 in 32,768 colour

VIDEO MODES

The graphics adapters usually operate in one of several different modes. A VGA card will, for example, operate in 'text mode' using either 80 or 40 columns, and in 'graphics mode' using 4,16, or 256 colours. The graphics adapter contains one more VLSI devices that organize the data, which produces the screen display. To determine the current video mode, one can read the machine's video mode byte stored in RAM at address 0449. this byte indicates the current video mode. The code snipnet below tests for current video mode.

```
E% = PEEK(&H449)
IF E% = 0 then mode$ = "40 X 25 mono (CGA/EGA/VGA)"
IF E% = 1 then mode$ = "40 X 25 colour (CGA/EGA/VGA)"
IF E% = 2 then mode$ = "80 X 25 mono (CGA/EGA/VGA)"
IF E% = 3 then mode$ = "80 X 25 colour (CGA/EGA/VGA)"
IF E% = 7 then mode$ = "80 X 25 mono (MDA/Hercules)"
```

MOBILE AGENTS FOR REMOTE MAINTENANCE TASKS: CLIENT DIAGNOSIS

WHAT IS AN AGENT?

According to Bret Sommer (1997). All researchers have no precise definition for the world agent. There are at least two reasons why it is so difficult to define precisely what agents are. Firstly, agent researchers do not 'own' this term in the same way as fuzzy logicians/AI researchers, own the term 'fuzzy logic' – agent is a term thats is used widely in everyday parlance as in travel agents, estate agents, to mention a few. Secondly, even within the software fraternity, the word 'agent' is really an umbrella term for a heterogeneous body of research and development. However, an agent can be referred to as a component of software and/or hardware, which is capable of acting exactingly in order to accomplish tasks on behalf of its users. Furthermore, due to multiplicity of roles that an agent can play, there is now a plethora of adjectives which precede the word 'agent', as in the following drawn only from Stan (1996) paper: search agents, report agents, presentation agents, navigation agents, role-playing agents, management agents, retrieval agents, domain-specific agents, analysis and design agents, etc.

Agents for the remote maintenance task are moving from considering the fundamental questions of what is intelligence and intelligent reasoning (El-Darieby, 1998), and how it can be manifested in a computer. Such maintenance agents typically may not have intelligence as their core; they are as intelligent as they need to be to complete the task they have been given. This is the belief held by Mennie (1998) when he stated that "…intelligence is in the eye of the observer" and that the intelligent behaviour of an agent arises as a result of its interaction with its environment.

DIFFERENT TYPES OF AGENTS

In a broad sense, the precepts of agents technology exist in many of the applications we use today and take for granted, (Sahai, 1997). For example, e-mail client is a type of agent. At a request, it goes about its business of collecting the unread e-mail from the mail server. However, there are several dimensions to classify existing software agents based on what they do and technology that underpins these agents.

Firstly, agents may be classified by their mobility (Krause, 1996) that is by their ability to move around some networks. This yields the classes of static or mobile agents.

Secondly, they may be classified as either deliberate or reactive (Dorigo 1996): they posses an internal symbolic, reasoning model and they engage in planning and negotiation in order to achieve coordination with other agents.

Thirdly, agents may be classified along primary attributes they should exhibit: A*utonomy*, *learning* and *cooperation* (Megadanz 1996). Autonomy refers to the principle that agents can operate on their own, without the need for human guidance, even though this would sometimes be invaluable. Hence, agents have individual internal states and goals, and they can act in such a manner to meet its goals on behalf of its user. Cooperation with other agents is paramount (Krause, 1996),

wherever there are multiple agents. Lastly, for agent systems to be truly 'smart', they could have to learn as they react/interact with their environment. These attributes led to the derivation of 3 types of agents: According to Neison M, (1998), *Collaborative agents* emphasized on cooperation and autonomy than on learning, *Interface agent* which is based on autonomy and learning and *Smart agent* that emphasized on cooperation and learning.

Fourthly, agents may sometimes be classified by their roles (Somer,1996), for example WWW information agents, which exploit Internet search engines, such as lycos, help manage vast amount of information.

Hence, seven types of agents are identified in this work:

Mobile agents
Static agents
Reactive agents
Collaborative agents
Interface agents
Smart agents
Information agents

MOBILE AGENTS: AN OVERVIEW

A mobile agent is a software or program that is able to migrate to some remote machine, (Mennie, 1998), execute some function or collect some relevant data and then migrate to other machines in order to accomplish another task. The basic idea of this paradigm is to distribute the processing throughout the networks, that is, send the code to the data instead of bringing the data to the code. According to Bieszczad et al. (1998), mobile systems differ from other mobile code and agent-based technology because increased code and state mobility allow for even more flexible and dynamic solutions. Telecommunication applications and network management are part of the broad range of application fields of mobile agent systems (Susilo et al., 1998).
To better understand mobile agents and their behavior, some traditional network architectures would be briefly looked into. The following figure illustrates the network behaviour of a typical client/server application.

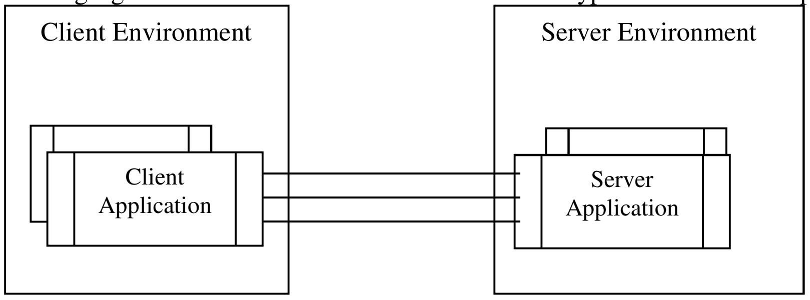


Figure 2.1: The typical client/server application communicates via requests and responses, which require a round trip trek across the network.

A client/server application typically consists of two pieces: a client piece and a server piece. Often, the client and server pieces are on separate machines and they communicate over a common network.

When the client needs data or access to resources that the server provides, the client sends a request to the server over the network. The server in turn sends a response to the request. This "handshake" occurs again and again in traditional client/server architecture. Each request/response requires a complete round trip across the network. Comparing the client/server architecture I just described to the mobile agent architecture illustrated in the figure below;

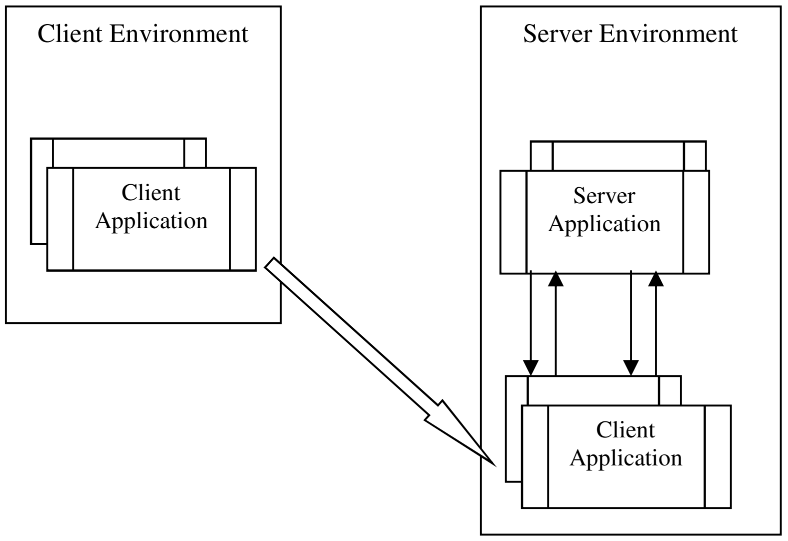


Figure 2.2: In the mobile agent architecture, the client actually migrates to the server to make a request directly, rather than over the network.

just as in the client/server architecture, there is a client piece and a server piece (Todd, 1998). The difference lies in how the two communicate. When the client in the mobile agent architecture needs data or access to a resource that the serve provides, the client does not talk to the server over the network. Instead, the client actually migrates to the server's machine. Once on the server's machine, the client makes its requests of the server directly. When the entire transaction is complete, the mobile agent returns home with the results.

A MOBILE AGENT'S TRAVEL HABITS

One distinguishing characteristic of mobile agent architecture is the mobility of the code (Carrez, 1999). However, while this characteristic is necessary, it alone is not sufficient. That is because the idea of moving code and computation to the location of the data and resources is not unique to the mobile agent architecture. In fact, such mobility has been a feature of many commercial databases for some time. In the database world, mobile code goes by the name of a *stored procedure*. A stored procedure is a piece of the client code that executes on the server. In some applications, the client piece of the application can dynamically upload stored procedures to the server. Once there, they can do their work and return the results of their calculations back to the client, (Di Caro, 1998).

The difference between the mobile agents and stored procedures is that stored procedures lack many of the features common to agents such as autonomy and flexibility. A more compelling reason is that once they are uploaded to a server, they belong to that server. If a transaction requires data or access to resources spread across servers, a stored procedure is forced to traverse the network just as the client piece did in the traditional model shown earlier. In other words, a stored procedure cannot migrate from server to server, dragging along the incomplete transaction or calculation with it. *A mobile agent can and does,* (Bill Venners, 2002).

CLIENT/SERVER ARCHITECTURES VS MOBILE AGENTS ARCHITECTURES

There are three very good reasons, why it might be necessary to move code from the client to the server, rather than have the two communicate over a network.
First, mobile agents solve the client/server network bandwidth problem. By moving a query or transaction from the client to the server, the repetitive request/response handshake is eliminated, (Bret Sommer, 1997).
Second, agents reduce design risk. Agents allow decisions about the location of code (client vs. server) to be pushed toward the end of the development effort when more is known about how the application will perform. The architecture allows for changes after the system is built and in operation, (Todd, 1998).
Third, agent architectures also solve the problems created by intermitted or unreliable network connections. Agents can be built quite easily that work "off-line" and communicate their results back when the application is "on-line", (Bill, 1997).

A MOBILE AGENT

A mobile agent is really a gestalt entity composed of two different pieces (Cheng et al., 1997). One piece is the code itself, which consists of the instructions that define the behavior of the agent. The second piece is the current state of execution of the agent.

Often, these two pieces are separate. For example, in a typical computer program, the code sits on disk while the executing state sits in RAM. A mobile agent, however, brings the two together. When an agent migrates to a new host, both its code and its state are transferred. Thus, the agent doesn't only remember what to do and how to do it, it remembers what it was doing before as well. When an agent migrates to a new host, both of these pieces must be captured and packaged. Todd (1998) referred the two pieces as resource (the code) and data (the state). The resource piece consists of the class files that define the agent. The data piece consists of a snapshot of the agent's state. More technically, it consists of a snapshot of the agent's data structures.

MOBILE AGENT

Mobile agent, as the name suggests, is the component within the architecture, which can migrate between client hosts, (Somers, 1996). They are mechanism by which the administrator exercise control over remote maintenance tasks. They are equipped with a set of goals specific to the user's task that describe the nature and limits of their functionality. In addition to the limits on functionality that users place on their mobile agents, it is probable that the agents themselves will encounter limits (in the form of security, authentication, validation and other restrictions) that exist within hosts.
The essential functions of the mobile agent will be defined by the specific diagnostic tasks that it is allocated. However, it has the following interactions with the architecture:

Mobile agent determines where to migrate to next by initially querying the Jini lookup service for a list of hosts of which it is aware. The mobile agent can then use this information to contact each host agents, in order to perform its designated tasks.
Mechanisms are employed to ensure that the mobile agent is not compromised during transit or within a host, and that its behaviors is consistent with the behaviour that it was given when it was launched.
Mobile agent possesses the characteristic of persistence; it uses migration as a mechanism to achieve longevity. In this way, it is not relevant on the host agent that launched it. This is particularly useful in nomadic computing, where the user is only connected to the network for short period of time.
Mobile agent may communicate not only with the host agent, but also with other mobile agents to achieve its goals.
It transmits the results of its findings and actions regularly to its user on the host. This way, the user can ensure that he mobile agent is functioning correctly and it can also help prevent the mobile agent from becoming too large (in terms of size) to move or to process.

THE LIFECYCLE OF A MOBILE AGENTS

Agents have a well-defined lifecycle. The figure below illustrates the four states that make up this lifecycle.(Todd,1998).

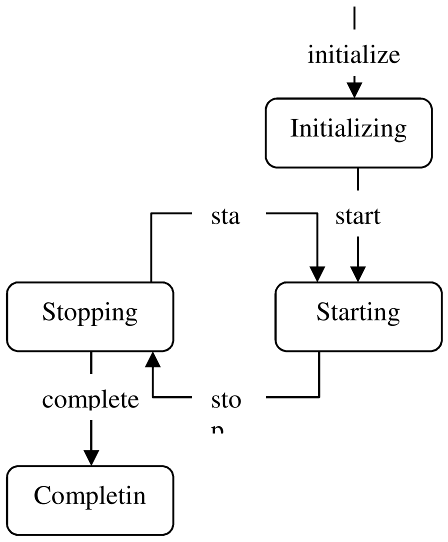


Figure2.3: An agent's Lifecycle

Initialize -	Performs one-time setup activities such as building initial data structures
Start -	Starts its calculations
Stop -	Stops its calculations, saves intermediate results, joins all threads, and stops
Complete -	Performs one-time termination activities

BASIC MOBILE AGENT MODELS

To make use of mobile agents, a system has to incorporate a mobility framework (Kotay et al.1994). The framework has to provide facilities that support all of the agent models:

Life-cycle model
Computational model
Security model
Navigation model
Communication model

For the life-cycle model, service to create is needed, destroy, start, suspend and stop, etc. agents. The computational model refers to the computational capabilities of an agent, which includes data manipulation and thread control primitives. The security model describes the ways in which agents can access network resources, as well as the ways of accessing the internals of the agents from the network, (White J.E, 1994). The communication model defines the communication between agents and between an agent and other entities (for example the network). All issues referring to transporting an agent (with or without its state), between two computational entities residing in different locations are handled by the navigation model.

MOBILE AGENTS SERVICES AND TOOLS

A variety of product and services companies offer agent-based solutions (Bret Sommer, 1997). Most of these services include the capabilities to move across the network and retrieve information at rapid perusal. Some providers of these kinds of agent services are PointCast, BackWeb and My Yahoo!. The FireFly network takes this model a step further by attempting to introduce people with similar interests in music and video through online chat and discussion groups.

On the tools side, no discussion of agents is complete without mention of General Magic and its Telescript technology. However, Java's widespread acceptance threatens to unseat Telescript as the agent-creator toolbox, (Lange, 1997).

JAVA THE AGENT-CREATOR

Mobile agents are powerful, versatile, and – possibly most important – fun to work with. Java provides an ideal implementation platform, furnishing tools that help streamline complex software applications. Java's Jini framework facilitates mobile agent application development, providing key features for distributed network programming. According to Bill Venners (2000), Jini is one such Java tool that allows the developer to build distributed applications with relative ease. Just as robots automate main aspects of manufacturing a computer, Jini automates and abstracts distributed applications' underlying details. These details include the low-level functionality, (socket communication, synchronization and so on) necessary to implement the high level abstractions (such as service registration, discovery, and use) that Jini provides.

Jini is a simple set of protocols layered on top of the Java Virtual Machine (JVM), via Remote Method Invocation (RMI). Among other things, Jini has been positioned as a "plug and work" technology. Solving the problem of bootstrapping into a network from a device is a significant accomplishment. But Jini has the potential to do so much more, such as facilitating both new approaches to system and network management. The Jini protocols and programming interfaces are:

Discover and Join: help an arbitrary device that enters the network.
Lookup: Creates a repository of available services
Object leasing: facilitating a fundamental "transaction" view of the network
Distributed events: extends the Java event model to work in a distributed manner.

MOBILE AGENTS FOR REMOTE MAINTENANCE

Network management has always been one of the most privileged demonstration fields of mobile agent technology, (Sahai, 1997). Network management means different things to different people. In some cases, it involves a solitary network consultant monitoring network activity with an outdated protocol analyzer. In other cases, network management involves a distributed database, auto polling of network devices, and high-end workstations generating real-time graphical views of network topology changes and traffic. In general, network management is a service that employs a variety of tools, applications and devices to assist human network managers in monitoring and maintaining networks, (Sahai, 1997).

The ISO has contributed a great deal to network standardization. Its network management model is the primary means for understanding the major functions of network management system.

Fault management

According to Yemini (1993), the goal of fault management is to detect, log, notify users of, and (to the extent possible) automatically fix network problems to keep the network running effectively. Because faults can cause downtime or unacceptable network degradation, fault management is perhaps the most widely implemented of the ISO network management elements, among which is remote maintenance.

The ability to take control of a client desktop from remote location serves to significantly reduce call service time. By remote monitoring the PC, administrations can gather correct information about its state and correct the problem directly – while on the phone with the customer.

The use of mobile agents for remote diagnosis addresses the problem of taking care of on-site devices that provides certain services at customer premises; for example, in a network for on-demand video delivery (Yemini, 1993). In such networks, it will be impossible to send a technician to test every faulty device, because there will be hundreds of thousands or more of them installed. A mobile agent will be sent instead. It may perform a suite of tests and attempt to repair the devices if possible. Only if that fails will a human operator be involved. The agents can incorporate machine-learning techniques, which may improve their future behaviour, (White et al., 1998).

CHARACTERISTICS OF DIAGNOSTIC MOBILE AGENTS

Diagnostic mobile agent's posses the following characteristics among others:

Autonomy. Once launched or activated with information describing the bounds and limitations of their tasks, **Diagnostic agents** should be able to operate independently of and unaided by their users.

Social ability. To effect changes or interrogate their environment (personal computers and their peripherals), **Diagnostic agents** must posses the ability to communicate with the outside world.

Reactivity. **Diagnostic agents** need to be able to perceive their environment and respond to changes to it in a timely fashion.

Proactivity. To help **Diagnostic agents** to be adaptive to (learn) new situation, they need to be able to exhibit proactivity, that is, ability to effect actions that achieve their goals by taking the initiative

Mobility. **Diagnostic agents** will able to move easily across the network. The underlying infrastructure should provide a language-level primitive that an agent can call to move itself to neighboring node or workstation.

Goal Oriented. They must be able to identify and monitor the status of any device in a particular node.

Persistence. Once a **Diagnostic agent** is launched, it should not be reliant on the system that launched it and should not be affected if that node fails. The concept of its mobility should give it the ability to 'survive' and to reach as many nodes as possible. This will be useful for the network administrators due to the fact that they can log on, launch an agent, log off and check later on its progress.

Fault tolerance. In the event of a network or server failure, it is often difficult for the client to reclaim situation and re-syncronise with the server because the network connection would have been lost. However, since the proposed **Diagnostic agents** do not need to maintain permanent connections and their state is centralized within themselves, failures would be easier to deal with.

THE DIAGNOSTIC SYSTEM ARCHITECTURE

The diagnostic mobile agent architecture comprises of two main components. The first component is the mobile agents themselves; that is, entities that move between network nodes to perform the diagnostic tasks. The second component is the mobile agent host(s), - static agent(s) – the service that provides the mobile agent's execution platform. They provide resources and facilities to the mobile agent. In a distributed environment, these can exist one-to-many agent hosts as well as one-to-many agents. To be active agent platform, a given node in the system has at least one active agent host.

It's envisaged that the mobile agent will possess the ability to migrate to the most appropriate or specified location and will comprise behaviour and some form of knowledge. The behaviour segment of the agent describes the reasoning ability of the agent and also its functionality, for example, hard disk test. The knowledge segment contains the 'intelligence' of the agent, that is, its experience, its goals and its beliefs. This segment also contains temporary information that the agent used to act autonomously before it is returned to the user for appraisal. Figure 3.1 below depicts the diagnostic system architecture while Figure 3.2 shows the agent framework components.

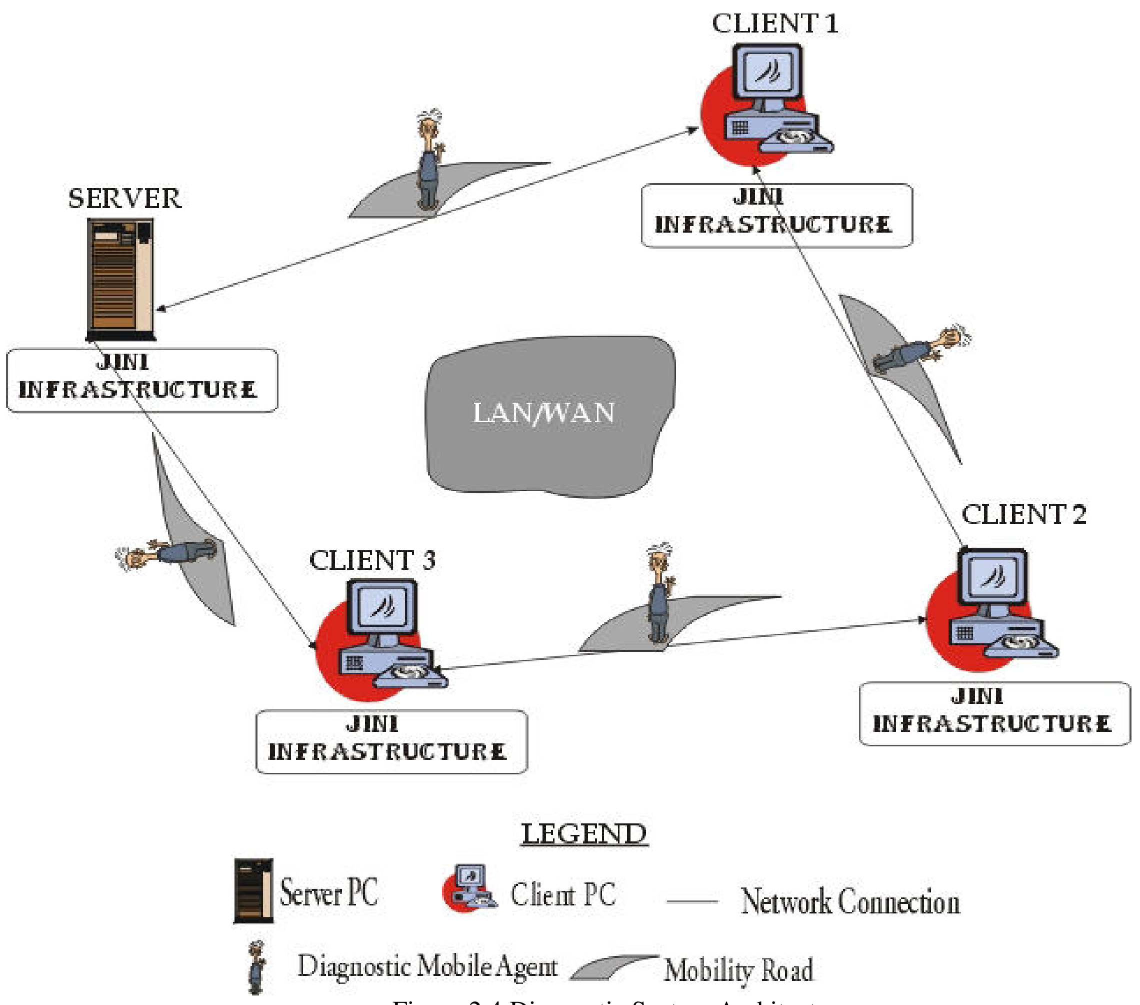


Figure 2.4:Diagnostic System Architectu

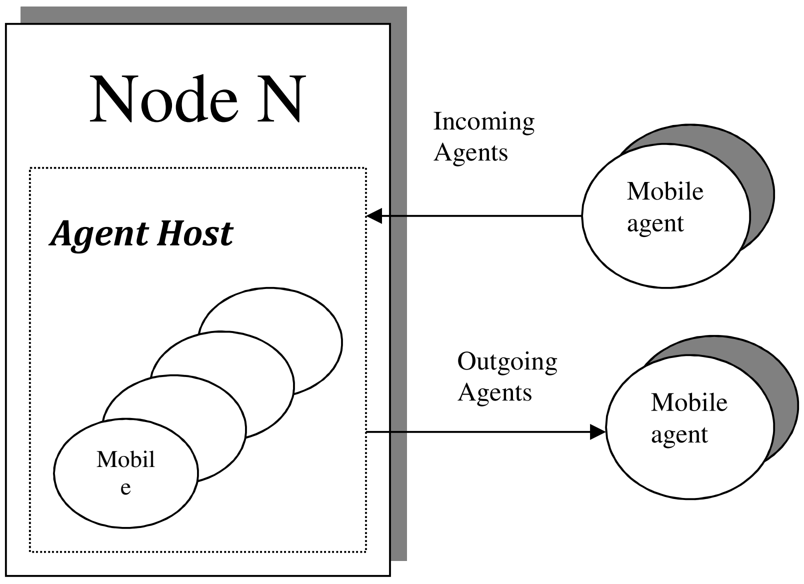


Figure 2.5: Diagnostic Agent Framework components

Figure2.6: Service Registration and discovery

These two components map quite nicely to the Jini model, (Jonathan et al., 2001). Jini at the highest level provides the infrastructure that enables clients to discover and use various services. In the context of this mobile agent framework, the agent host(s) provides Jini services. The diagnostic agent is the Jini client.
Jini services register with one or more Jini lookup services by providing a service proxy for prospective clients. In turn, client query the lookup service(s) for particular services that might be of interest.

CORE ARCHITECTURE AGENTS

Each of the core agent within the architecture (Mobile agent, Interface agent and Host agent) comprise the following components:

- Jini Mobility Infrastructure
- A core behaviour set that describes the basic functions available to the agent
- An extensible behaviour set that describes the services (diagnostic tests) that the agent can perform, and,
- A free-form knowledge for storing working or diagnosis information, which is preserved across migration.

This section takes each core agent of the diagnostic mobile agent architecture and describes its general activities and purpose within the infrastructure.

USER INTERFACE AGENTS

The user interface agents provides a level of abstraction for the user away from the details of the mobile agent architecture, This agent may be described as a *personal assistant* who is collaborating with the user in the same work environment (Kotay et al., 1994). It's essentially an interface agent that performs the following tasks:

- It launches mobile agent on behalf of the user and tasks its progress and position
- It provides mobile agent with a communication point through which they can return the results of its tests. User interface agent may need to posses local mobility to allow for node failures, since it needs to be executing even when a user is disconnected from the system.
- It organizes and pre-processes test reports return from mobile agents into a form that is suitable for the user. This may involve filtering out replicated information, presenting urgent test information to the user quickly, rendering information using the tools that the user is most familiar with.
- User interface agents provide the user with a window onto their status, their results, and the mobile agent architecture.

HOST AGENTS

According to Lange et al.(1997), mobile agents migrate between computing locations called *hosts*. Host agents are static agents that exist within the hosts to provide the resources mobile agent needs to work. They are also responsible for handling the mechanics of packaging an agent and moving its resource and data pieces from one host to another. The Figure 2.7 below illustrates how this occurs.

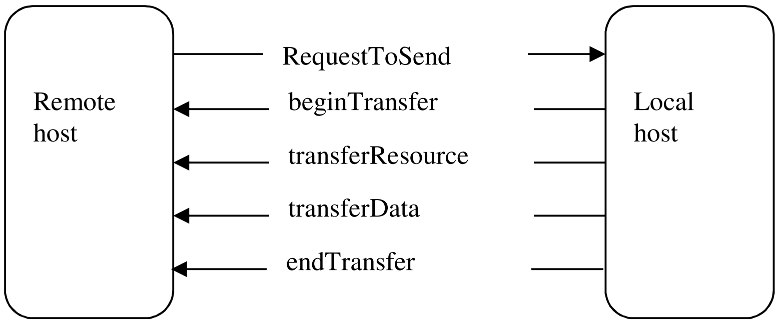


Figure 2.7: Agent Hosts communicate with each other to move the agent along its way

There are two hosts in this figure: the *remote host* and the *local host*. The local host is responsible for orchestrating the transfer. The remote host is responsible for initiating the transfer – possible at the request of the agent itself. The agent initially resides on the remote host. When the series of communications is complete, the agent will reside on the local host.

AGENT HOSTS CONSTRUCTION

The first step in building the agent host is to create a remote interface, the service template that agents will look for via the Jini lookup service (Krause, 1996). The *AgentHostRemoteInterface* provides one method, *acceptAgent()*, which agents call to travel to the implementing agent host:

Public interface AgentHostRemoteInterface extends Remote{
Public void acceptAgent(AgentInterface ai) throws RemoteException; {

The second step in building the agent host is to provide an implementation of this remote interface that is the actual Jini service. I provided a *MobileAgentHost* class that implements the *AgentHostRemoteInterface was provided.* Figure 4.4 below shows the class diagram for *MobileAgentHost.*

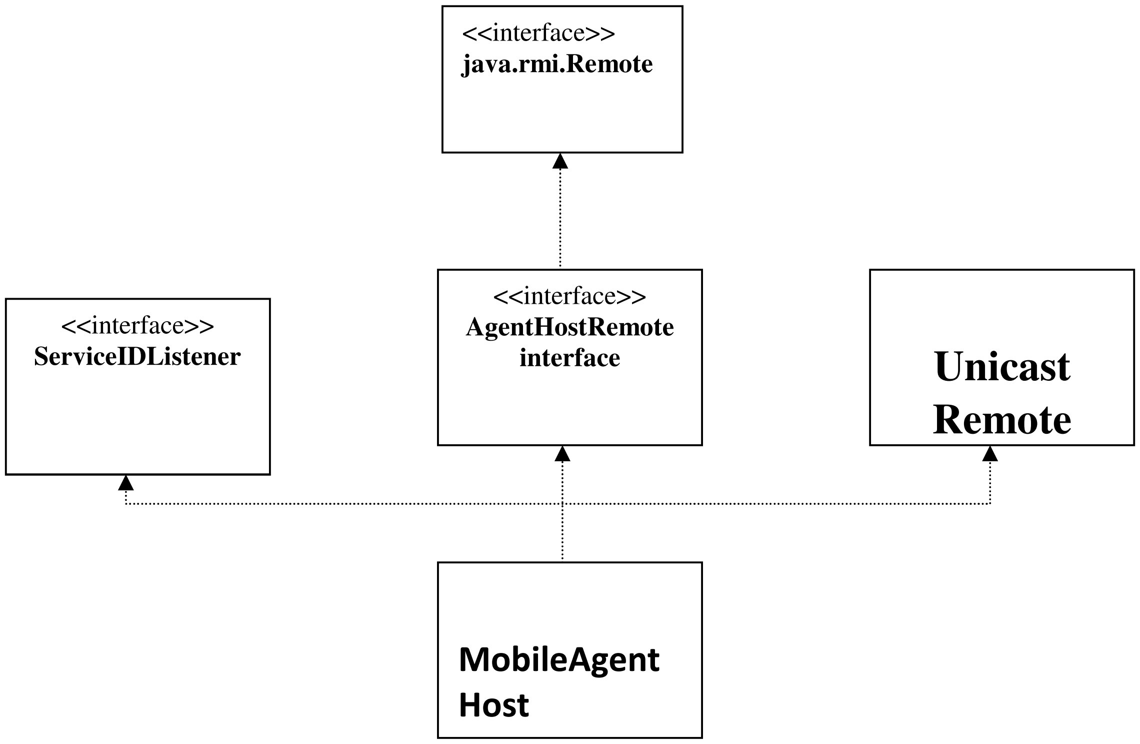


Figure2.8: Mobile agent host class diagram

The class extends the *java.rmi. server* package's *Unicast Remoter Object* class, which allows clients to obtain a remote reference and call its methods. The *Mobile Agent Host* also implements the *Service ID Listener* interface, which is passed a unique *Service ID* object via the *Service ID Notify()* method when the service first registers with a Jini lookup service.

This *Mobile Agent Host* constructor takes the *agent Object* reference and passed to arriving agents via the *do works()* method. The constructor itself performs 2 basic functions: it creates a *Look Up Discovery Manager* to locate a Jini lookup service and, a *Join Manager* to add this lookup service to the Jini service federation.

In the *accept Agent()* method's implementation, the *mobile Agent Host* binds an incoming agent to an *agent Thread*, which is turn calls the *do works()* method; passing the *Lookup Discovery Manager* and agent object references. In this implementation, a new thread is created for each arriving agents.

MOBILE AGENT CONSTRUCTION

The first step in building the agent is to create an interface for the agent. In this architecture, An *Agent Interface* that extends the *Serializable* interface was created. The *Serializable* interface marks the implementation as a *serializable* entry that can be sent across the wire. The *Agent Interface* consists of a *do work()* method that is called consists agent arrives on a given host. This method takes two parameters. The first parameter is a reference to the *Lookup Discovery Manager* maintained by the current host. The agent uses this reference if and when it decides to look for new service providers such as when it wants to travel to a new agent host. The second parameter is an optional object parameter, which contains data necessary for the agent to complete its jobs.

The second step in agent implementation is providing an *Agent Interface* implementation for which an abstract *Mobile Agent* class was created. This class's constructor builds a service template that locates services of type *Mobile Agent Host Interface*. It also provides three additional methods: *do work(); Move to Random Host();* and *get Mobile Agent Host*()

To perform an agent-specific task, sub-classes override the abstract *dowork()* method.

When the agent want to move, sub-classes called the *movetoRandomHost()* method which perform the following three steps:

Gets a list of the currently available mobile agent hosts with a call to *getMobileAgentHost()* method.

Randomly select a host from this list

Moves to new host by calling the *accept Agent()* method. If the call on the selected host fails it select a new host.

To obtain a list of currently available agent hosts, sub-classes called the *get Mobile Agent Host()* method. This process require the following steps:

Call *get Registrars()* to obtain a current list of lookup services.

Iterate through each lookup service to find services that match desired template; in this case, *Agent Host Remote Interfaces*. The *My MAH Service Template* object, a *Service Template* class instance, passed to the lookup() method initializes the *Mobile Agent* constructor.

Each matching service is added to a vector of *Agent Host Remote Interfaces*.

Figure 2.9 below depict the interactions between the *Mobile Agents*, *Mobile Agent Host*, and the Jini lookup service.

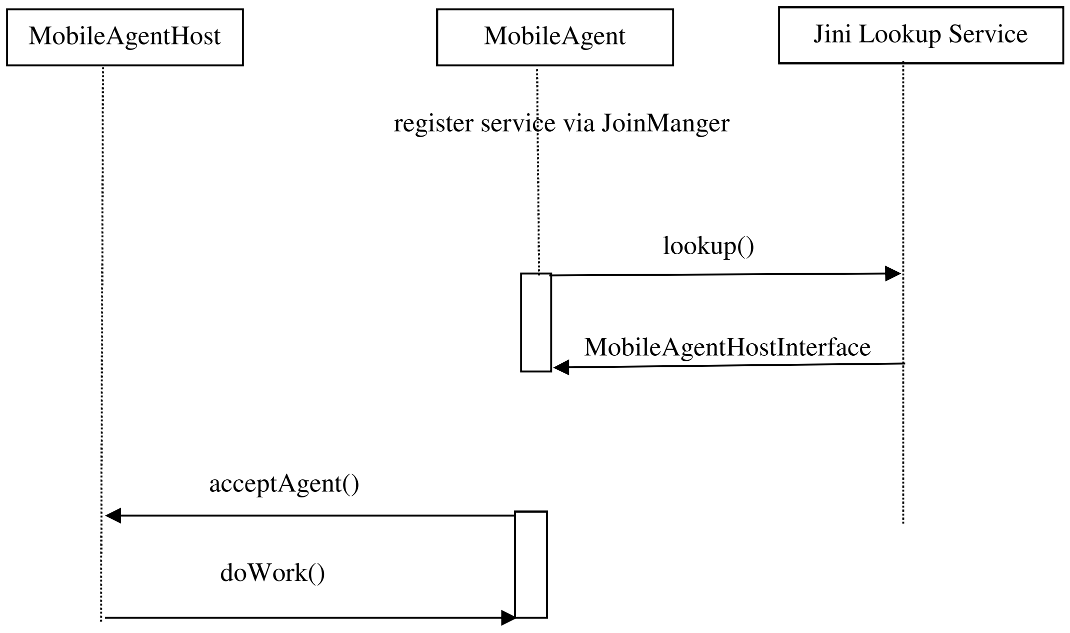


Figure 2.9: Sequence diagram for interactions between the Mobile Agent Host, Mobile Agent, and Jini lookup service.

The final step in building the agent is to create a concrete *Mobile Agent* implementation. In the implementation *Route Map Agent* extends *Mobile Agent*. This agent dynamically travels to various agent hosts, logs its routes and calls diagnostic service routine. In this implementation, the object passed to the *do Work()* method by the *AgentHost()* is the local host name. The agent records this name in its route map. Figure 3.0 below depicts the *Route Map Agent()* class diagram.

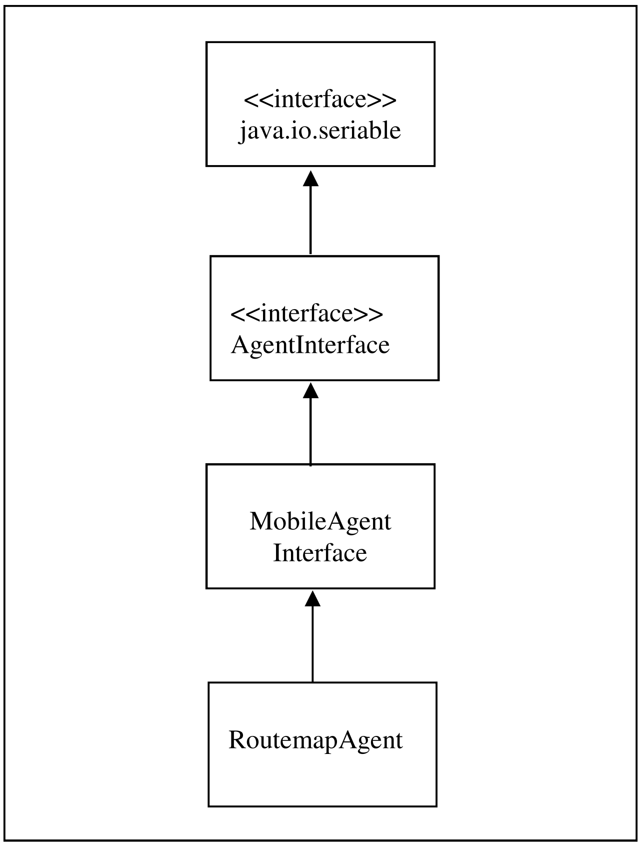


Figure 3.0: Route Map Agent class diagram

In the designed architecture, all agent activities start from the server. In order to set up the environment, the following steps are taken:

Start an HTTP (HyperText Transfer Protocol) server for clients to download Jini class files on the node where the Jini lookup service will run:

Java –Jar toolsk.jar –port 8081 –dir c:\agent\lib -Verbose

Start an HTTP server that will provide the agent class file, this server must start on every node in which an agent initializes:

Java –Jar toolsk.jar –port 8082 –dir c:\agent –Verbose

Start an HTTP server that will provide the agent host class file; this server starts on every node in which an agent host runs:

Java –Jar toolsk.jar –port 8083 –dir c:\agenthost –Verbose

Start the RMI (Remote Method Invocation) activation deamon:

rmid –J-Djava.security.policy=policy.all
where policy.all is given as
grant {
permission Java.security.AllPermission "","";
};

Start the Jini lookup service:

Java –Djava.security.policy=c:\agent\lib\policy.all,
-jar reggie.jar http://ola: 8081/reggie-dl.jar c:\agent\lib\policy.all
c:\agent\lib\logfiles public

where's **ola** is the name of the server machine on which the Jini HTTP server started.

Once the environment is set up, the agent hosts on the client PCs are started, before instantiating the diagnostic agent on the server. On instantiation of the agent hosts, each in turn, registers with one or more Jini lookup services. After starting the agent systems and instructing the server to start agent activities, the mobile agent query the Jini lookup service and retrieve the service proxy object, which contains the registered agent hosts; before the mobile agent is given full autonomy to manage its own affairs.

In addition, during instantiation the mobile agent displays the list of available agent hosts and allows its sender to specify those that it should visit. With this behavioral information the mobile agent perform its activities and determines dynamically the next machine to visit. It then sets up a "client" connection routine which is the sub-component of Jini mobility infrastructure, to establish the link between the current machine and its destination machine; where the Jini mobility infrastructure on the machine behaves as the "server" for these clients to establish a communication channel based on Transmission Control Protocol/Internet Protocol (TCP/IP) socket. Once a connection is established, the agent moves through this channel (mobility road) to the destination machine where it is instantiated again to perform its diagnostic tasks. On reaching the end of specified agent host, the mobile agent travels back to the sender to give report. The agent creates some service components as it performs its activities, which are necessary for the agent system to generate reports of the diagnosis.

One interesting feature of the diagnostic mobile agent is its ability to perform diagnosis on client that has been switched off. With the Wake-on-LAN technology incorporated into present Network Interface Cards (NICs), they remain active in low-power mode after the client is switched off and continuously monitor the flow of packets on the network. So, if the adapter observes a wake-up-frame packets sent to its own Machine Access Control (MAC) address from the mobile agent, it turns the client's computer on. This capability permits the intelligent diagnostic agent to perform its functions even when nobody is around with the client system.

SYSTEM ANALYSIS

ANALYSIS OF MOBILE AGENTS APPLICATION

The following scenarios can be used to illustrate the analyses of how a client server and mobile agent can diagnose a fault on a network. The administrator may used conventional client/server technology to send diagnosis request from the server to several other client machines connecting to it. For example mcafe total perfection system licence. During the time, the administrator need to stay at the central terminal to control the connection between the server and the client machines in the LAN.Instead of the above framework, mobile agent could be sent out from the server machine. to various clients for the diagnosis.
If each arrow in the figure represents one message sent, we can see that using a mobile agent actually saves two messages in a network of three servers, compared with using client/server.
In general, if there is n server, (n-1) messages can be saved using mobile agents.
Mobile agents are one of the most prominent technologies believed to be plying an important role on future fault diagnosis of computers on network systems.
Besides providing a very flexible approach for information gathering.
The mobile agent application analyses based on the following:
SPEED: The speed of mobile Agent will be determined so as to know faster platform out of the two in electronic commerce.
DISTANCE: The distance cover will also be used tom determined the total distance that each platform will need to cover for a particular transaction.
FAULT DISCOVERY This will be used to discover the capable platform that can use in fault diagnosis.

HOST TRANSACTION

The distance covered to visit the host 1is x 1
The distance covered when returned to the local machine is x1
The total distance covered to perform this is 2x1
For 2 host.
X2

Fig 3.3 hosts X1
X1=Distance used, Yi = Distance between Host! And 2
X2 =Distance between Host 2 local machine
Therefore, the total distance covered for 2-host transaction is X1+ Y1+X2.
Hence, for host 3
The total distance covered will be.
X1+Y1+Y2+X3
Where.
Y2= distance between host 2 and host 3
X3= distance between local machine and host 3
Host 4
The total distance covered will be
X1 + Y1 + Y2 +X3 +X4
Where

Y3= distance between host 3 and 4
X4 =distance between host 4 and local machine and host
Host 5
The total distance covered is
X1 + Y1 + Y2+X3 + x4+ X5
Where
Y4 = Distance between host 4 and host 5
Therefore, the speed is a function of time and distance; the speed of mobile agent will be compare with that of client server because of the two reasons mentioned below.

1. When both used to cover same distance in one host transaction, mobile agent arrived earlier.

Mobile agent reduced total distance covered to perform a transaction of two or more host.
Therefore, $\text{SPEED} = \dfrac{\text{DISTANCE}}{\text{TIME}}$

Since the distance cover by mobile agent and the time is reduced and both are the function of speed then, mobile agent can be compare with that of client server.

ANALYSES OF CLIENT SERVER APPLICATION

The client server as it illustrated in section 3.1 is show in fig 3.1 a when a client server is sent to fetch information in an electronic commerce to visit different client machine is very tedious because it involves establishing connections each time a server wants connections to a machine A disconnecting from the machine before connecting with second machine. Therefore the performance criteria like speed, time distance covered and fault discovery can be used to analyse the client server.

SPEED: The client server speed can be determined and compare with that of mobile agent because, from various researches and experiment carried out by people. It was affirmed that client server consumes more time.

DISTANCE: Distance cover by the client server can be determined so as to know the totals distance that each platform May require to cover for a particular transaction.

FAULT DISCOVERY: The number of fault that each platform discovery in two or more host was also determined.

For host 1:
Let X1 =Distance between the local machine and host 1
Then X1` Distance will be covered when queried and X1 distance will be covered when replied.
Therefore, total distance covered by the client ser4ver will be.
TD1=2X1
For 2 hosts:
Let X2 =Distance covered from local machine to host 2
Then, the total distance covered for the transaction is TD2= 2X1+2X2
For host 3
Let X3 = distance covered from local machine to host 3
Therefore, the total distance covered will be
TD3= 2X1 +2X2+ 2X3
TD4= 2X1 +2X2 +2+X3 +2XTD5 = 2X1 + 2X2 + 2X3 +2X4 +2X5.

IMPLEMENTATION AND RESULTS

This application was developed using both client server and mobile agent platforms and simple experiment was carried out where some criteria like Speed, Time Distance and fault discovery were used. In this chapter, an explanation of the implemented experiment was given. The chapter also contains observations that are inferred from result of the experiment

DIAGNOSTIC INFRASTRUCTURE IMPLEMENTATION

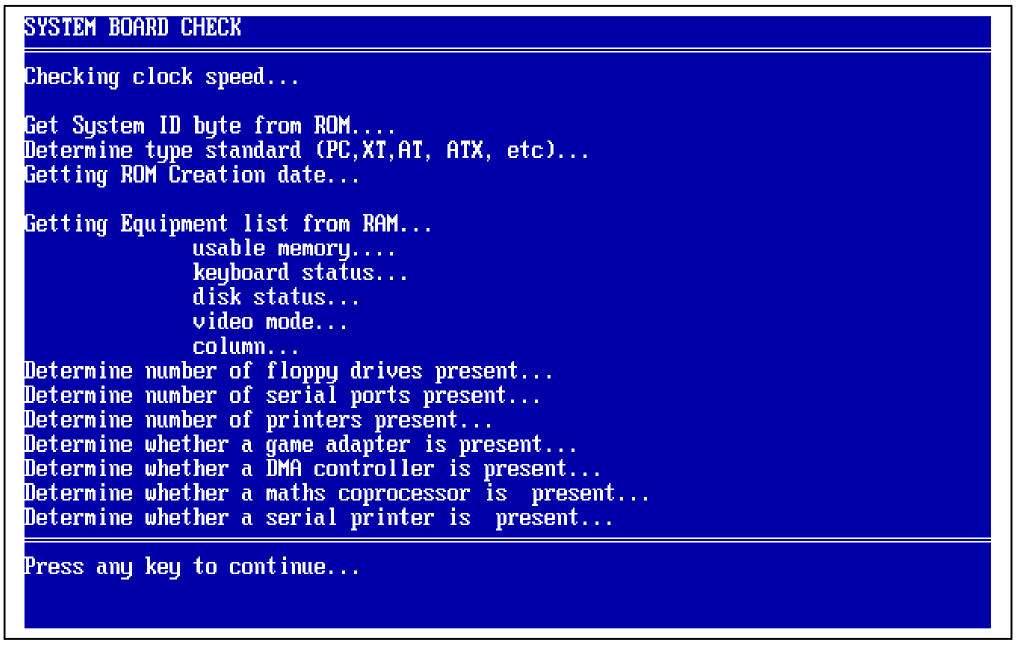
The Fault Diagnostic application was implemented using java programming language with the support of the jini API. System implements mobile agents and client server to check the status of available devices within a microcomputer system in a network environment. The agent performs various diagnostic tests on each component of client machine and report back to the server, so as to aid administrator or technician in taking quick good decision in troubleshooting and repair of any faulty system across network.

DIAGNOSING WITH CLIENT SERVER/MOBILE AGENT

This application performs the fault diagnosing using client server/mobile agent and the report of diagnosing was summarized.

SYSTEM INFORMATION

DMA directly queries hardware to identify the exact device where possible such as CPU, BIOS name and version, detailed fixed disk and floppy information, video information, and port information, printer connected. Information is extremely detailed to include items such as hard drive manufacturer name and
model number. In addition, it gives the amount of conventional (base) RAM present, alerts user if PC games adapter, DMA controller, maths coprocessor, and Internal Modem are present. Figure 1 below shows sample system check activity.

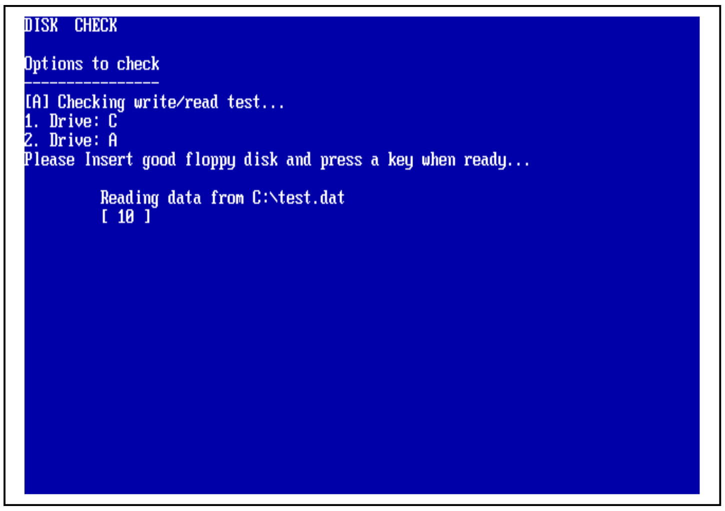


```
SYSTEM BOARD CHECK

Checking clock speed...

Get System ID byte from ROM....
Determine type standard (PC,XT,AT, ATX, etc)...
Getting ROM Creation date...

Getting Equipment list from RAM...
              usable memory....
              keyboard status...
              disk status...
              video mode...
              column...
Determine number of floppy drives present...
Determine number of serial ports present...
Determine number of printers present...
Determine whether a game adapter is present...
Determine whether a DMA controller is present...
Determine whether a maths coprocessor is  present...
Determine whether a serial printer is  present...

Press any key to continue...
```

Figure1: Sample System Board Check

HARD DISK TESTING

DMA performs Read, Write, and Seek tests on any fixed disk. Also displays and edits physical parameters, partition parameters, and CMOS parameters on any drive. This information is extracted from the drive itself, not a database like other diagnostics software and does not have to be updated It finds any physical or electronic defect on the drive or the controller allows for relocating Track 0 on supported IDE drives. Safe Write Test allows user to perform exhaustive write testing without destroying any data on the drive. Figure 2 below depicts sample disk check.



```
DISK  CHECK

Options to check
----------------
[A] Checking write/read test...
1. Drive: C
2. Drive: A
Please Insert good floppy disk and press a key when ready...

         Reading data from C:\test.dat
         [ 10 ]
```

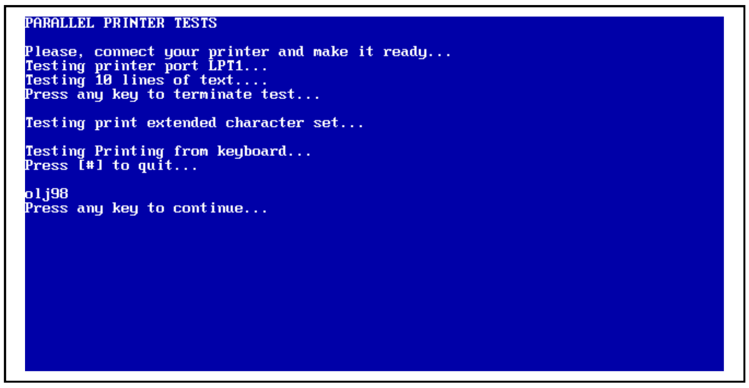
Figure 4.2: Sample Disk Check

FLOPPY DISK TESTING

Accurate testing of any portion of a floppy drive, including read, write, format, safe write, and butterfly seek tests. Tests all media formats up to 2.88mb and includes a user-defined option for higher media formats.

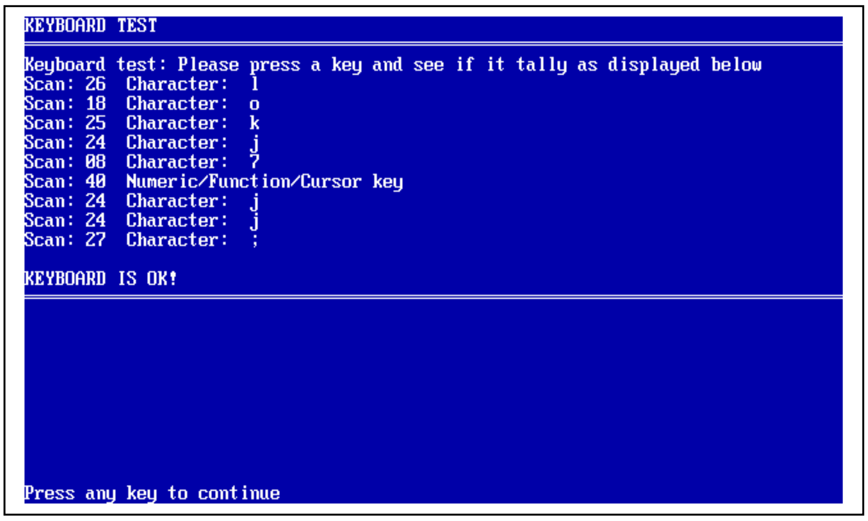
PERIPHERAL TESTING

Complete an accurate testing of mouse, joystick, keyboard, printer, and sleep button. The sample tests are shown below:



```
PARALLEL PRINTER TESTS

Please, connect your printer and make it ready...
Testing printer port LPT1...
Testing 10 lines of text....
Press any key to terminate test...

Testing print extended character set...

Testing Printing from keyboard...
Press [#] to quit...

olj98
Press any key to continue...
```

Figure 3: Sample Printer test



```
KEYBOARD TEST

Keyboard test: Please press a key and see if it tally as displayed below
Scan: 26  Character:  l
Scan: 18  Character:  o
Scan: 25  Character:  k
Scan: 24  Character:  j
Scan: 08  Character:  7
Scan: 40  Numeric/Function/Cursor key
Scan: 24  Character:  j
Scan: 24  Character:  j
Scan: 27  Character:  ;

KEYBOARD IS OK!

Press any key to continue
```

Figure 4: Sample Keyboard test

SERIAL AND PARALLEL PORT TESTING

DMA provides the most extensive and accurate port testing available, far surpassing the capabilities of DOS or Windows based diagnostics. Any possible error will be detected and identified. All ports tested, regardless of IRQ or I/O port assignment. All lines are tested on the external tests with the included loopback connectors. Instantly identifies UART capabilities. Also test FIFO capabilities of any serial port having these capabilities. It detects the interrupt a serial or parallel device is actually using, regardless of how it is believed to be configured.

MODEM TESTING

DMA tests modems in connected or non-connected mode, including major AT modem commands. Retrieves modem's information and type directly from the modem's chipset.

DISPLAY/MONITOR TEST

This carries out various checks and adjustments on variety of PC displays including basic text mode display (80 X 25 monochrome), CGA, EGA, VGA and SVGA colour types. Various tests that are performed include:

- **Alignment test**: This test displays a series of concentric circles and it is used for adjusting height, width, vertical linearity, and horizontal linearity. A sample alignment test is shown below:

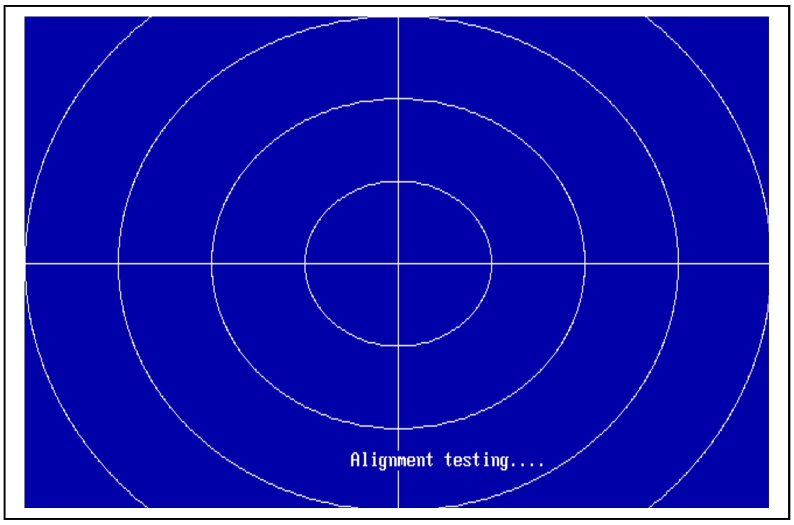


Figure 5: Sample Alignment testing

Grid test: This displays a grid lines and it is used to check monitor convergence.

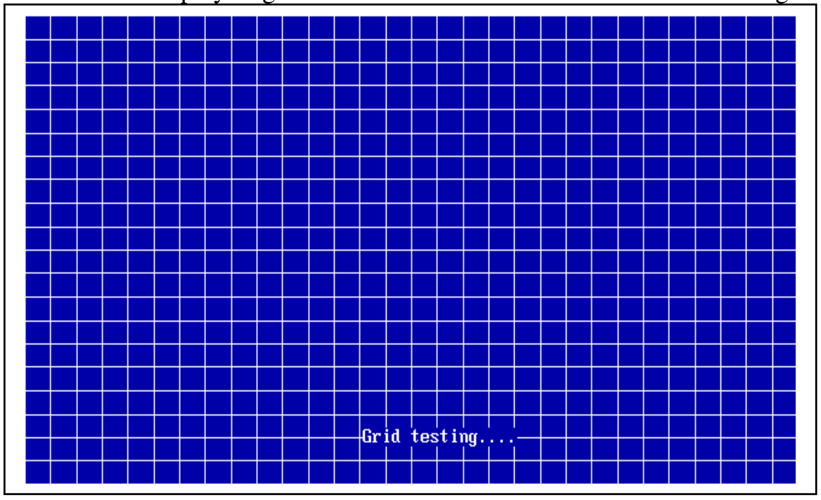


Figure 6: Sample Grid test

- **Dot test**: This displays a matrix of single-pixel dots. It can be used to check focus (all dots should be the same size).

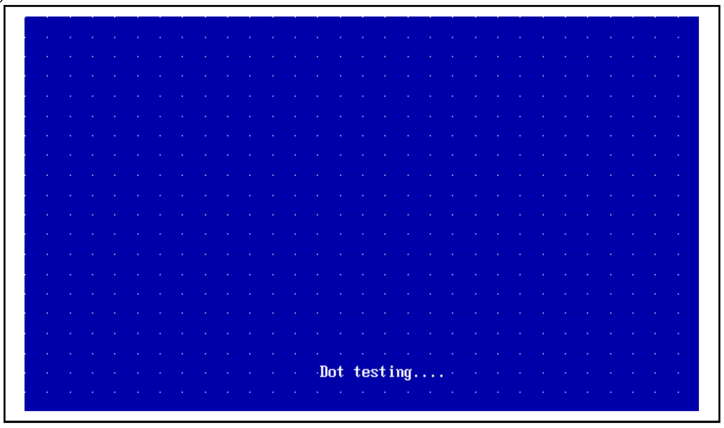


Figure 7: Dot testing

- **Colour test**: this displays 16 colour bars and can be used for performing clour adjustments.

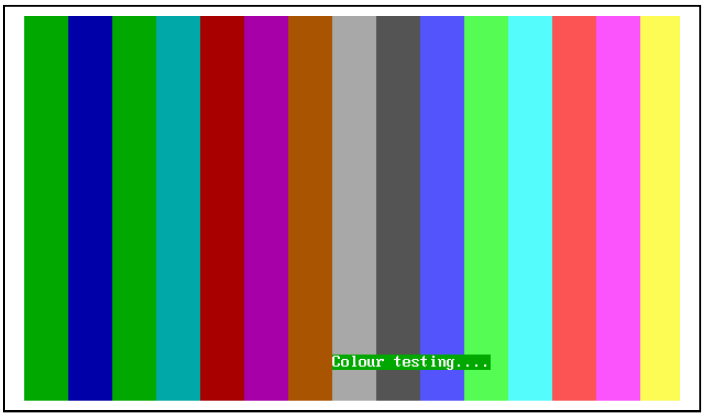
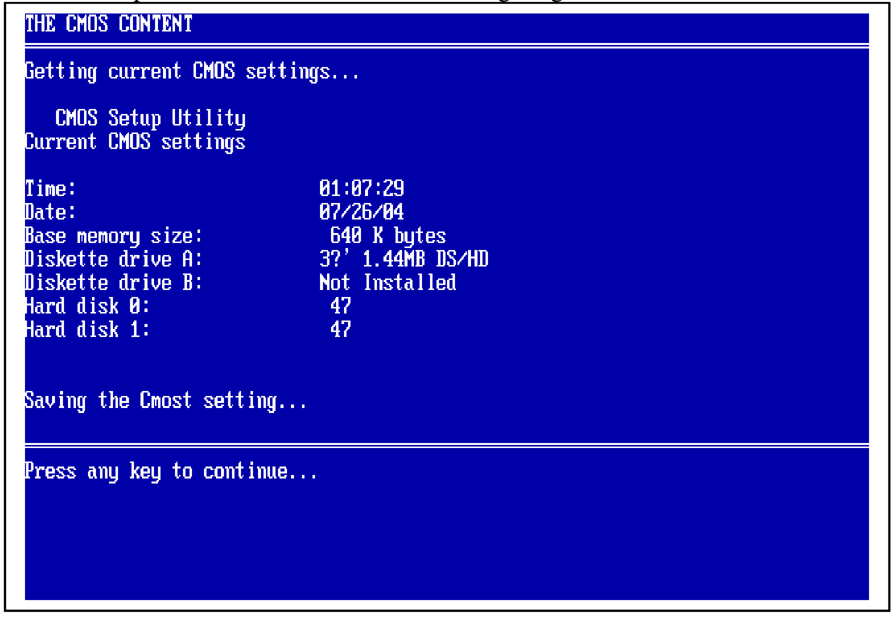


Figure 8: Sample Colour test

- **Text display**: This displays a checkerboard of text characters (ASCII 32 and 219 respectively).

RUN CMOS SETUP
Display and edit CMOS settings. Very useful for setting up older machines for which the original setup diskette is no longer available. Sample CMOS content retrieved during diagnostic routine is shown below:



```
THE CMOS CONTENT

Getting current CMOS settings...

   CMOS Setup Utility
Current CMOS settings

Time:                           01:07:29
Date:                           07/26/04
Base memory size:                640 K bytes
Diskette drive A:               3?' 1.44MB DS/HD
Diskette drive B:               Not Installed
Hard disk 0:                     47
Hard disk 1:                     47


Saving the Cmost setting...

Press any key to continue...
```

Figure 9: Sample CMOS content

REPORTING
Diagnostics results are directed to the server, where it can be displayed or printed. Sample result is shown in Appendix B.

DATA ANALYSIS AND TESTING
The analysis from the demonstration shows that mobile agent is faster than client server because of the time used to cover a specific distance.

- Mobile agent use time of 1.5s to cover a distance of 10m.
- Clients server use the time of 2.5s to cover a distance of 10m.
  Therefore, to get their speed,
- Speed of mobile agent= distance/time=10/1.5 = 6.667m/s
- Speed of client server = distance / time 10/2.5 = 4.0m/s
  In the diagnosing information system, 1-10 hosts were used for the demonstration. To determine

1. Distance needs to cover to visit certain host (host)
2. Time consume by each platform of perform same transaction

Samples data used as distance are:
X1 = 10, X2 = 15, X3 = 20, X4 = 25, X5 = 30

Y1 = 15, Y2 = 20, Y3 = 25, Y4 = 30.
Below data were generated for the two

TABLE 1 CLIENT SERVER AND MOBILE AGENT

HOST	DISTANCE COVERED BY CLIENT SERVER dx(m)	DISTANCE COVERED MOBILE AGENTS xy(m)	TIME USED CLIENT SERVER t(s)	TIME USED BY MOBILE AGENT t(s)
1	20	20	5	3
2	50	40	12.5	4.5
3	90	65	22.5	9.75
4	140	95	35	14.25
5	200	130	50	19.5

From the data generated in the table above, the following inferences were drowned.

- Mobile agent covered less distance to perform same transaction than client server
- Mobile agent consume less time to cover same distance than client server.

Therefore, mobile agent has a greater speed than the client server.

FAULT DISCOVERY OF BOTH PLATFORMS IN FAULT DIAGNOSING

From the demonstration of electronic commerce applications, it was discovered that mobile agent discovered more product in two or more host (host).
In the demonstration, 1-5 hosts were used and 10 product were searched for.
Below data were generated from the demonstration

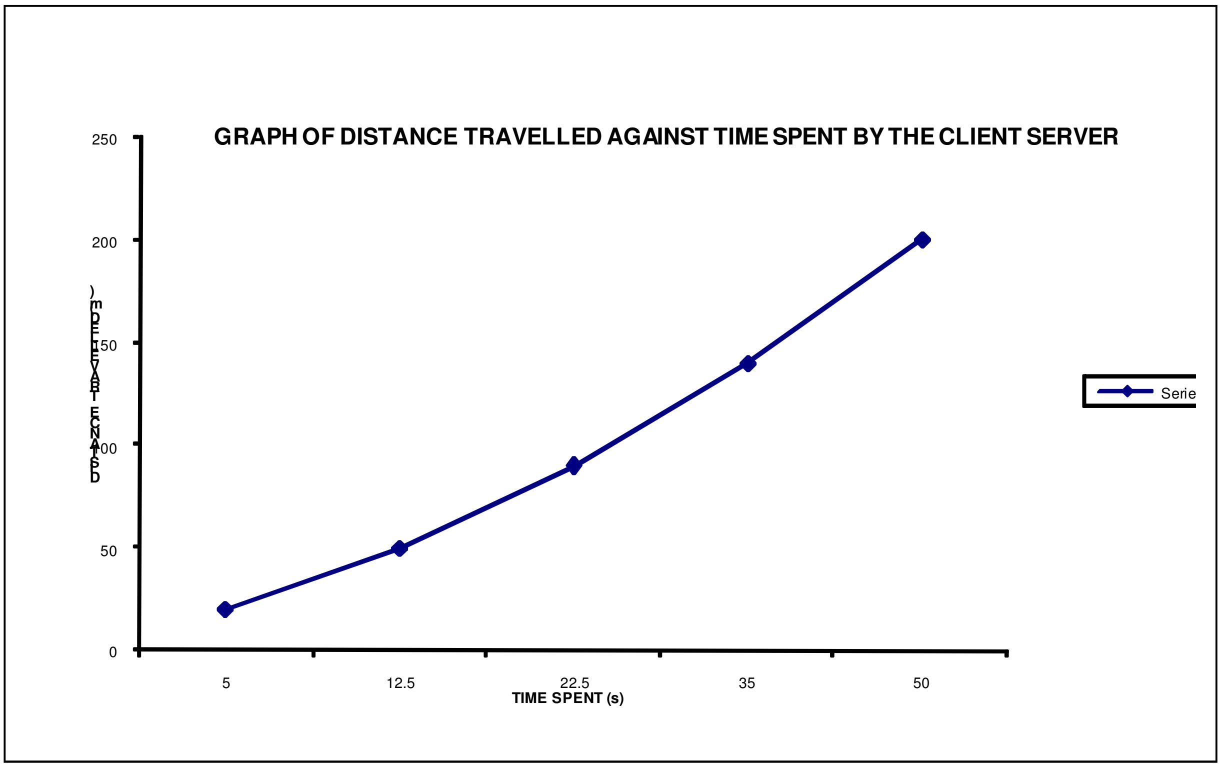
TABLE 2 PRODUCT DISCOVERY OF BOTH PLATFORM

HOST	CLIENT SERVER PRODUCT DISCOVERY	MOBILE PRODUCT DISCOVERY
1	10	10
2	7	9
3	6	9
4	6	8
5	5	7

COMPARISON ANALYSIS

Fault diagnostic system is designed using the object-oriented paradigm because the concept of the object is useful to describe agents. There are two main types of objects in the system, namely Agent and Launch server.

Graph of Time spent to cover a certain Distance by the client server in Fault diagnostic



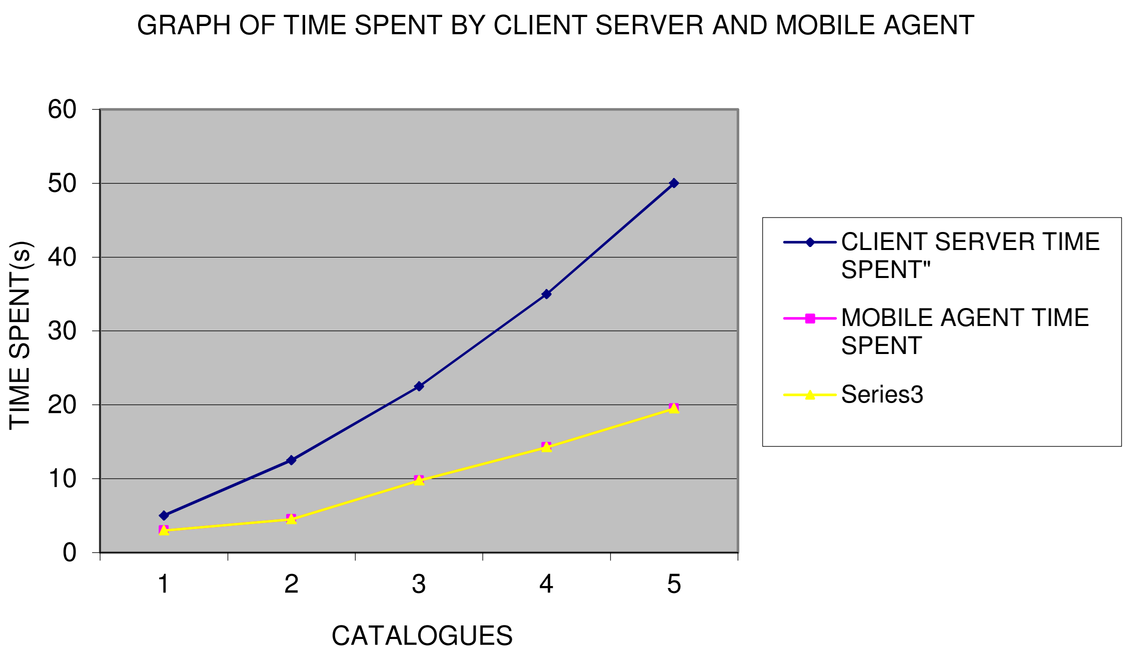


Figure 10: The graph of time used to cover a specific distance in fault diagnosing

The above graph shows the time used to cover a particular distance by the client server in fault diagnosing. Therefore, time used to cover a particular and the distance cover at a particular time can be determined.

Graph of time spent to cover a certain Distance by the mobile agent in a fault diagnosis.

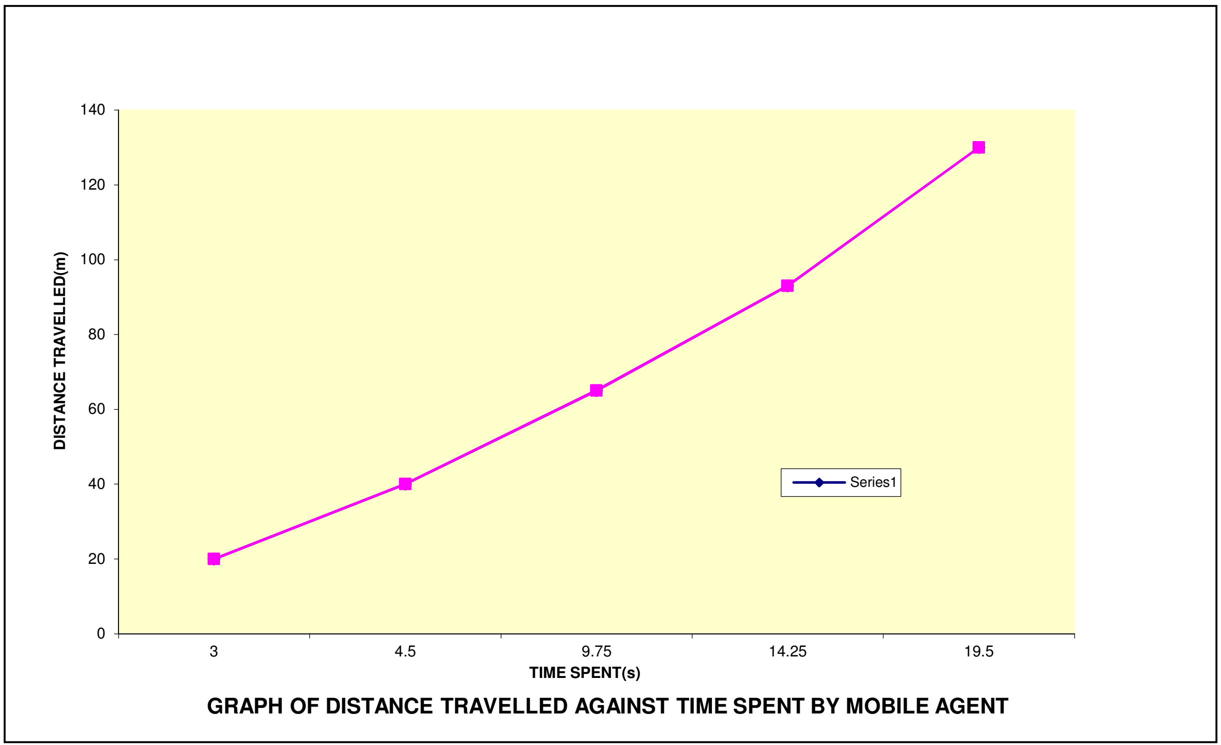


Figure 11: Graph of time spent by mobile agent to cover a certain distance

The above graph shows the time used to cover a particular distance by the mobile agent in fault diagnosing. Therefore, time used to cover a particular and the distance cover at a particular time can be determined.

The graph showing the comparison of the distance covered in a particular fault diagnosing by both platforms.

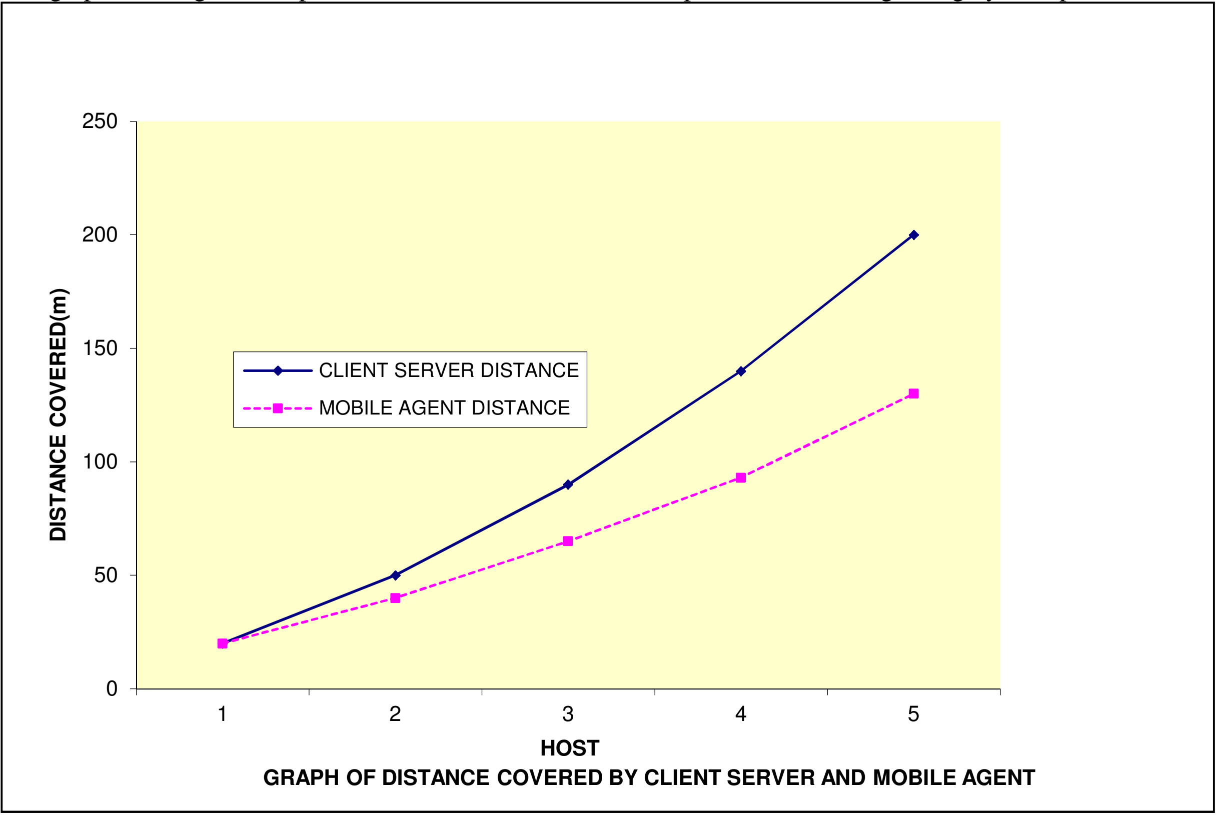


Figure 12: The comparison of distance covered by both platforms

The above diagram show distance covered by both platform and from the graph it can be deduce that client server covers more distance then the mobile agents in performing same transaction. Therefore, mobile agent is cost effective in term of distance needed before a transaction is completed.
The graph showing the comparison of the time used in a particular transaction by both platforms.

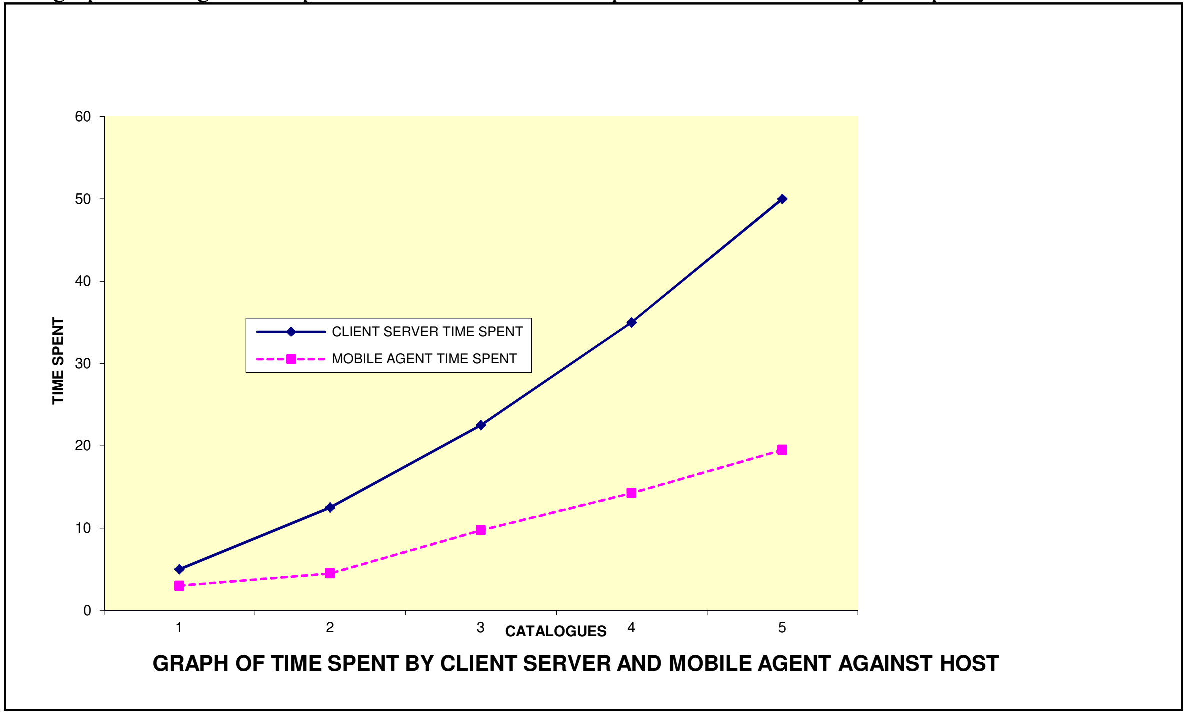
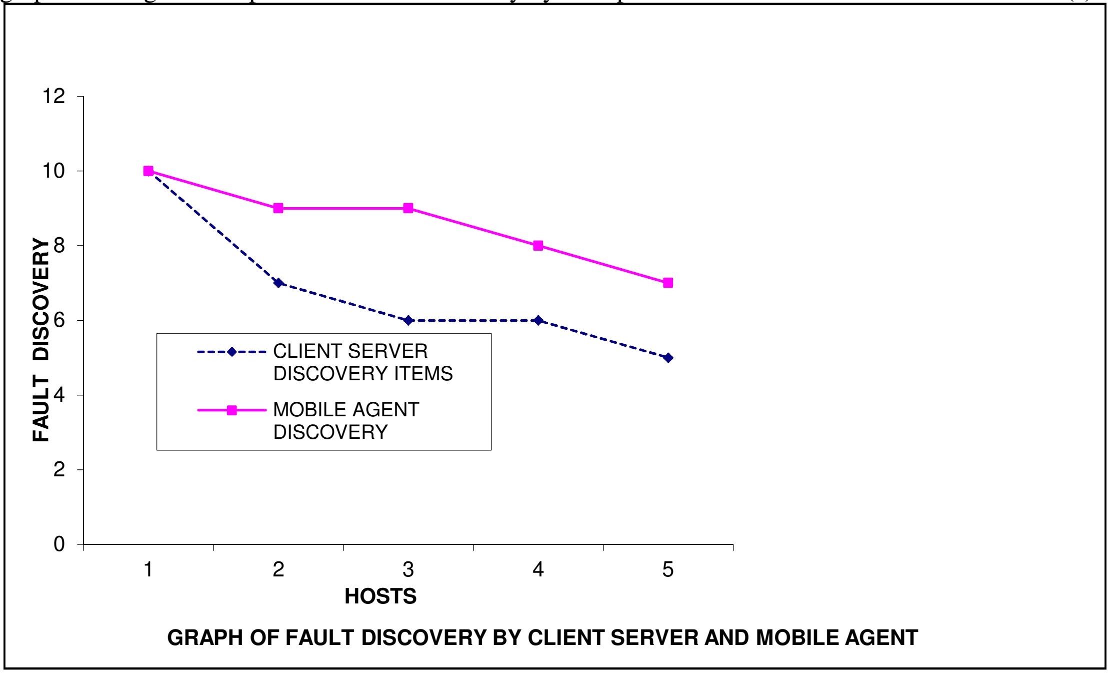


Figure 13: The comparison of mobile agent and client server time

The above graph shows the comparison of mobile agent and client server time used. From the above it is seen that mobile agent used less time than client server to cover same distance and to perform same transaction.

The graph showing the comparison of fault discovery by both platform, when visit same number of host or host
The graph showing the comparison of fault discovery by both platforms when visit same number of host(s).



The above shows how each platform are able to discover item (s) when visit same number of host. From the above graph it can be deduced that both platforms (mobile agents and client server) were able to discover same number of fault when visit one host but, mobile agent were able to discover more fault than client server in tow or more hosts in a fault diagnosis system.

DIAGNOSTIC AGENT INSTALLATION ROUTINE

Firstly, the installation files are deflated from the packaged file to 'Setup folder'. At run dialog box, type this command: D:\MOBILEAGENTPACK.EXE and pres Enter key.

Installation for the diagnostic mobile agent (DMA) requires these steps:

(1) Install environment class files, so as make available JINI lookup service files.

(2) Install the Printer drivers so as to configure the agent to available current printer on the server. The following commands are used to install environment and printer files: Type on the command line:

C:\Agent>SetupEnvi.exe and press Enter key
C:\Agent>SetupPrinter.exe and press Enter key

where *SetupEnvi.exe* installs environment class files, SetupPrinter.exe installs the printer driver files.

The next step is to setup mobile agent itself on the machine where it is going to be instantiated.

C:\Agent>SetupAgent.exe and press Enter key

where *SetupAgent.exe* installs agent files

Finally, the AgentHost files are setup on each client machines, using the following command:

C:\AgentHost>SetupEnvi.exe and press Enter key
C:\AgentHost>SetupHost.exe and press Enter key

where *SetupHost.exe* installs agent hosts files.

To run the system, the server environment is configured to start the Jini lookup service, using the command below:

C:\Agent\Lib>StartEnviron and press Enter key

Secondly, each client AgentHost is started to register with the Jini federation, using the following command:

C:\AgentHost>StartAgent OlaAgentHost

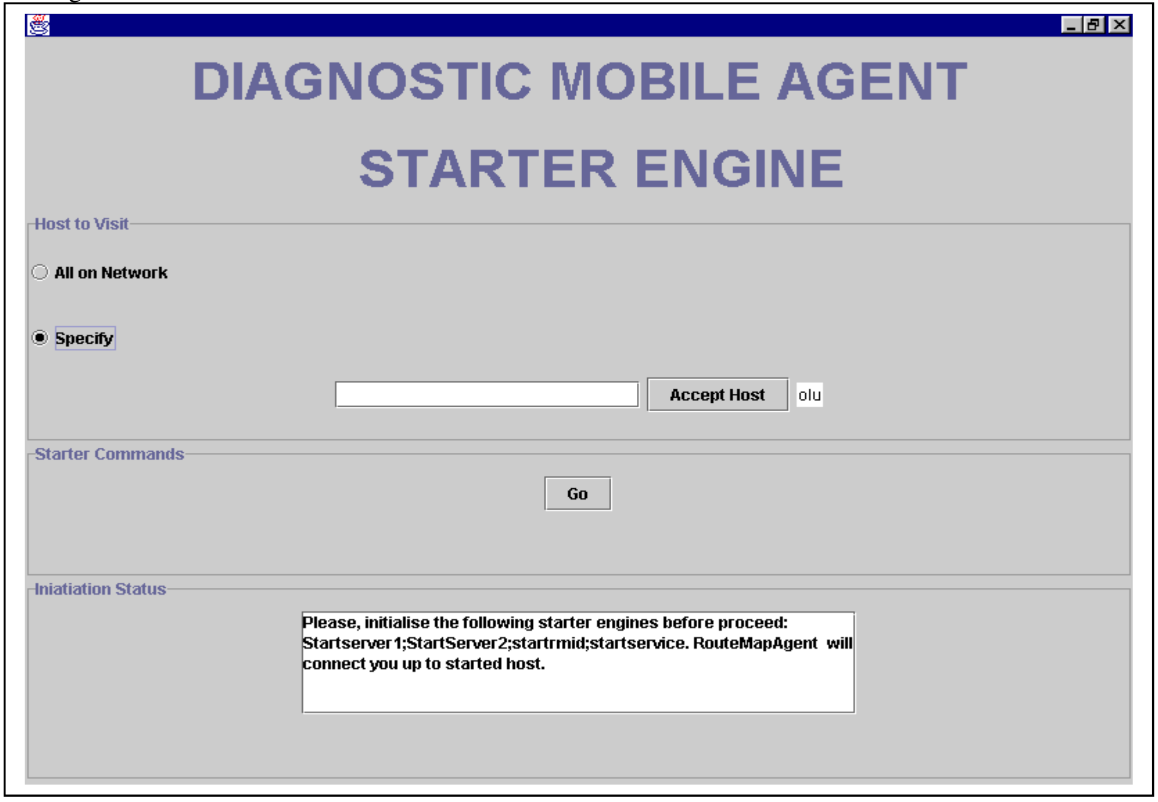
where *OlaAgentHost* is the hostname of the client machine.

Finally, the diagnostic mobile agent is started and launched to the network. This command is used:

C:\Agent>StartAgent DMA

where *DMA* is the name of the mobile agent.

The figure below shows the starter window:



Please, initialise the following starter engines before proceed: Startserver1;StartServer2;startrmid;startservice. RouteMapAgent will connect you up to started host.

Figure 15s: Mobile Agent Start Window

ACKNOWLEDGEMENT

I owe my profound gratification to the God Almighty who in His Infinite Mercy grants me the grace to start and complete this publication.

I appreciate Mr ALABI S.A the author of my publication as well as my supervisor DR. ADEOSUN O.O. who had helped me in the reading, correction and typing of this publication also INTERNATIONAL INSTITUTE FOR SCIENCE TECHNOLOGY AND EDUCATION(IISTE) for accepting my publication.

Thanks and God bless.

CONCLUSION

In this research work, we study the mobile code paradigm, which is a collection of remote evolution, code on demand, and mobile agent as an alternative to the convention client/ server paradigm. We examine fault diagnosis of micro computers.

The work also discusses features like speed, Distance traveled and time taking was used for the analysis. This work discusses system implementation and testing software installation, the test environment and the application were developed

This project has presented a new approach to monitoring the status of computers in a network environment. The project attempted to show that mobile agents were able to systematically and dynamically diagnose various components and devices connected to wide area network. At a more general level, It discussed the use of mobile agent in managing networks autonomously, with an intelligent architecture designed for the implementation of (remotely) maintaining resources on the network. The Jini's plug-and-work service was employed to develop the mobility engine, which provides fast and quick connections of devises on network. However, this project work has really designed to be of help to technicians and network administrators, but improvements are still needed to make it available for repairing fault components.

REFERENCES

Aderounmu G.A, Adagunodo E.R. and Akinde A.D., 2002 Diagnostic Mobile Software Agent for Microcomputer System. A Journal of Nigerian of Engineering Management, Vol. 3dNo. 2, April-June, 2002.

Ajulo Ojo, 2001 Computer System Maintenance, a project submitted to the department of Industrial Mathematics and Computer Science.

Andrzej Bieszczad, Bennard Pagurek, 1998 Mobile Agents for Network Management, IEEE publication, 1998

Baldi, M., Gai, S. and Picco, G. P., 1997 Exploiting Code Mobility in Decentralized and Flexible Network Management, First Int'l Workshop on Mobile Agents Mobile Agents '97, Berlin, Germany, April 7-8, 1997.

Baldi, M. and Picco, G. P., 1998 Evaluating the Tradeoffs of Mobile Code Paradigms in Network Management Applications, *20th Int'l Conf. on Software Engineering (ICSE '98),* Kyoto, Japan, Apr. 1998. Bieszczad, A. and Pagurek, B., 1997 Towards plug-and-play networks with mobile code, *Proc. of the Int'l Conf. for Computer Communications ICCC '97,* Cannes, France, Nov. 19-21, 1997.

Bieszczad, A. and Pagurek, B., 1998 Network Management Application-Oriented Taxonomy of Mobile Code, *Proc. of the EEE/IFIP Network Operations and Management Symposium (NOMS '98),* New Orleans, Louisiana, Feb. 15-20, 1998.

Bill Venners (*JavaWorld* April/May 1997) IBM's agent technology in a two-part Under the Hood series

Bill Venners (*JavaWorld,* February 2000)" Locate Services with the Jini Lookup Service,

Bret Sommers (*JavaWorld*, April 1997) Find out more about agents and IBM's Aglets technology with the detailed article "Agents: Not just for Bond anymore," penned by

Carrez F.,1999 Agent Technology and its Application. Alcatel Telecommunication Review. Ist Quarter, pp 67-77

Case, J. D. et al., 1990 Simple Network Management Protocol, RFC 1157, May 1990. [32. Case, J. D. and Levi, D. B., SNMP Mid-Level-Manager MIB, Draft, IETF, 1993.

Cheng, D. T. and Covaci, S., 1997 The OMG Mobile Agent Facility: A Submission, in Rothermel, K. and Popescu-Zeletin, R., Eds., Mobile Agents, Springer-Verlag, 1997.

Chess D., Grosof B., Harrison C., and Tsudik G., 1995 Itinerant Agents for Mobile Computing, IBM Research Report RC 20010, 1995.

Complete listing of *JavaWorld*'s Jiniology column:

Decker, K. S., Distributed Problem Solving, 1987 A Survey, *IEEE Trans. on Systems, Man, and Cybernetics,* vol. 17, no. 5, Sept. 1987, pp. 729-740.

Deitel H,M, and Deitel P.J. ,1998 Java How to Program, Prentice Hall

Di Caro G. Dorigo M., 1998 Mobile Agents for Adaptive Routine Proceedings. Third-first Hawaii International Conference on Systems

Dorigo, M., Maniezzo, V. and Colorni, A., 1996 The Ant System: Optimization by a colony of cooperating agents, *IEEE Trans.on Systems, Man, and Cybernetics-Part B,* vol. 26, no. 1, 1996, pp. 1-13. El-Darieby, M., 1998 Intelligent Mobile Agents for Network Fault Management, Technical Report SCE-

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar