Mobile Agent-Based Network Administration System

Stephen O, Ohwo.[1], Jonathan O. Erihri O.[2], Maryann Anya [3]
Department of Computer Science, Delta state polytechnic, Ogwashi-Uku, Nigeria [1,2,3]
steveohwo@yahoo.com[1,2], steveohwo@gmail.com[3]

Received: 12-JUNE-2024; Reviewed: 21-NOV-2024; Accepted: 12-DEC-2024
https://dx.doi.org/10.4314/fuoyejet.v9i4.2

Abstract—Managing, monitoring, and maintaining computer networks are complex process, especially when dealing with diverse networks. Automating and streamlining network management across heterogeneous networks remains a significant challenge. To address this, our research proposes a framework that leverages on mobile agents to perform network management tasks. This framework accommodates the limitations of various network devices that cannot run mobile agents. The objectives of this study is to create a network management framework using mobile agents to investigate the utility of equipment, internet protocol (IP) routers, asynchronous transfer mode (ATM) switches and several management stations in real heterogeneous network. Also, the framework can be utilized on existing networks. Methodology: previous frameworks were reviewed and a framework was proposed, thereafter a mobile Agent-based network administration system (diagnostic and search) was developed. Experimental results show that the mobile agents developed can efficiently execute certain network management tasks, particularly search and diagnostic tasks, which can pinpoint the root cause of network failures by exploring alternative paths to gather additional data and the model have 92.78% accuracy performance from analysis.

Key words - Network management, Mobile agents, Network diagnosis, Framework

1 INTRODUCTION

Network management encompasses various processes and activities to ensure network reliability, security, and performance. These processes include network monitoring, configuration management, fault management, performance management, security management, troubleshooting, and network optimization. Traditional network management systems have limitations, such as centralized management (Liotta et al., 2012) and vendor-specific solutions. To address these limitations, researchers have explored the use of mobile agents for network management. Mobile agents offer decentralization, flexibility, and automation capabilities, making them suitable for efficient network management. Various research projects, such as MAMAS (Mobile Agents for the Management of Applications and Systems) proposed in (Bellavista et al., 2023), JAMES (Silva et al., 2021), and the Network Management and Artificial Intelligence Laboratory at Carleton University, have developed mobile agent-based solutions for network management, demonstrating their potential for improving network management efficiency and performance. Novel architectures inspired by simple life organisms have been proposed. One such architecture is called ECOMOBILE (Rossier and Scheurer, 2020). It uses mobile agents to execute task objectives, but these agents are not by themselves network tasks. They have a life, compete with each other, exchange and take or leave task objectives at any time. This architecture offers an interesting way to regulate its mobile agent population while achieving network management tasks. ANTNET (Di Caro and Dorigo, 2018) is architecture in that field. It was introduced at first to use mobile agents for adaptive routing. However, it has inspired a lot of research (Pagurek *et al*. 2020). The common point of this research is the accomplishment of complex objectives using simple mobile agents. On the concept of proximity, a research group ((Liotta *et al*., 2022) has studied efficient ways to place mobile agents on a network combining both mobility and remote monitoring. Monitoring is one part of network management and may be used for fault management as well.

Recently in (Bohoris and Liotta, 2023) a performance management system based on mobile agents for virtual home environment has been proposed. However, this system is limited to performance management. New approaches using active nodes and lightweight agents, such as Weaver (Koon-Seng and Stadler, 2021) have been proposed in the literature. Likewise, Rayan *et al*., (2023), proposed Network management platform based on mobile agents. The new management platform architecture was based on ontology-driven mobile agents. Papavassiliou *et al.* (2022), proposed Mobile agent-based approach for efficient network management and resource allocation: framework and applications. They adopted efficient integration and adoption of mobile agents and genetic algorithms in the implementation of a valuable strategy for the development of effective market based routes for brokering purposes in the future multi-operator network marketplace. The proposed genetic algorithm provides a kind of stochastic algorithm searching process in order to identify optimal resource allocation strategies. The agent-based network management approach represents an underlying framework and structure for the multi-operator network model, and can be used to facilitate the collection and dissemination of the required management data, as well as the efficient and distributed operation of the algorithm. It also presents some numerical results to assess the

*Corresponding Author: teveohwo@yahoo.com
Section B- ELECTRICAL/COMPUTER ENGINEERING & RELATED SCIENCES
Can be cited as:
Ohwo. S.O., Erihri J. O., Anya M. (2024). Mobile Agent-Based Network Administration System. In FUOYE Journal of Engineering and Technology (FUOYE**JET), 9(4), 565-574.** https://dx.doi.org/10.4314/fuoyejet.v9i4.2

performance and operation effectiveness of our approach, by applying it in some test case scenarios.

These systems are highly-scalable but, in general, they cannot be utilized on existing networks since the majority of commercial routers do not permit the execution of user-supplied code. However, workarounds can be used, for instance, by attaching to each router a single-board computer (Koon-Seng and Stadler, 2021) or by using shared proxies. The first objective of this study is to create a network management framework using mobile agents to investigate the utility of them in real networks. Even though the concept of using mobile agents for network management has been considered before, it is the first time that such framework is implemented and tested using a heterogeneous network (containing, among other equipment, internet protocol (IP) routers, asynchronous transfer mode (ATM) switches and several management stations) in various environments. The proposed framework can also be used to manage elements that cannot receive mobile agents. The second objective of the study is that the framework can be utilized on existing networks.

However, Automating and streamlining network management across heterogeneous networks remains a significant challenge. To address this, our research proposes a framework that leverages on mobile agents to perform network management tasks. This framework accommodates the limitations of various network devices that cannot run mobile agents.

2. RELATED WORK

2.1 THE FRAMEWORK

The proposed framework recommends assigning one or two agents per network management task, which helps minimize inter-agent communication overhead. This approach avoids using multiple small or dependent agents, which can lead to inefficiencies. Research has shown that utilizing a single mobile agent for a task increases response time but reduces overall network traffic (Rubinstein, 2021). A performance evaluation of a real-world configuration task (Boyer, 2022) compared the use of a single mobile agent to parallel mobile agents for a task, providing valuable insights into the trade-offs involved. In fact, communication and synchronization between multiple agents slows the whole task to a point where a single agent performs better in both areas. It is therefore impossible to give an optimal choice for every topology, network size and management task. Our choice to use few mobile agents for one task is based on (Boyer, 2022) and on our motivation to limit the network load and agent building complexity. For tasks where there are few dependencies between each device, we suggest using multiple instances of the same mobile agent to divide the task. Network management is done using network tools already available. The advantage of using existing protocols such as SNMP is pointed out by (Zapf, 2020). They are already widely supported and implemented in network devices and limit the effort involved in building a network management framework. Also, mobile agents tend to use these tools more efficiently.

Management table: The management table is the only knowledge of the network that the framework provides. Any other knowledge is taken from the network as needed by mobile agents. This table keeps links between network elements and network management stations. One important experiment, associations in this table were static, meaning that one management station was bind permanently to one or many elements. The association is based on proximity; proximity may be determined by a wide selection of factors. In our case, it is simply the number of hops between the station and the element. Static associations do not mean that an element is always managed by the same station. It only tells the mobile agent the preferred management station for a particular element. The framework could use a dynamic update for this table and a way to adapt to network modifications made on topology. Liotta et al., (2022) offers interesting ideas on how this aspect could be improved. For optimization purposes, these tables are installed on each management station, freeing the mobile agent's memory to save bandwidth. This also allows local optimizations when it is not clear whether a central element must be managed by one station or another depending on the point of view.

Network Management Interface: Uniform interfaces are key parts of many mobile agent systems (Gavalas, 2021). However, our framework is not tied to uniform interfaces. This lets us introduce two kinds of mobile agents, general agents and specialized agents. The general agent will mostly use uniform interfaces, managing the network with limited functionality. The specialized agent is able to do a lot more tasks and use specialized features. Stationary agents are used to implement network management code that has to be dynamic, but may be totally inefficient to move with mobile agents. By dynamic, we mean that they could keep a state, be modified easily; keep local information in cache for fast and efficient retrieving. This code is moved once and stays permanently on the station. Stationary agents implement a set of uniform management interfaces. The management table that keeps references between management stations and elements also keeps a set of network management abilities for an element. Such abilities could be an operating system application programming interface (API), a protocol like SNMP or any other way to manage an element. These stationary agents look like interface agents found in (Timon, 2023) but fill a wider range of functionalities and utilities. Intelligence: Mobile agents need a great load of intelligence to be able to manage networks of heterogeneous devices. Although the framework uses uniform interfaces, it is still difficult to give agents sufficient intelligence to let them manage these networks confidently. Some research tends to use artificial intelligence or collective intelligence (White, 2020).

Our focus was to use an expert system, but the framework is not limited to a specific form of intelligence. The network tasks implemented using our framework aim to use proven procedural instructions that are best implemented by an expert system. Security and fault-tolerance: For networks where agents should move on user stations, the framework suggests, but does not yet implement, letting only the approved mobile agent's code to get back from user stations. For confidentiality purposes, mobile agents should give up sensible information from the network before entering a user station. Quotas may also be used to counter flooding. Since the framework is on top of any network management system, it does not interfere with these systems and is not needed for management. Fault management mobile agents described later are able to tell if the network management system is faulty, but they cannot recover completely from such a failure. ***Global view of the framework:*** Two logical networks are present: the management network and the normal network. The management network is essentially the management stations and the links between them. The normal network is the part assigned to useful applications. Both logical networks may be the same, have some devices and links shared or be completely different networks. In Fig. 1, we see a general view with a network device that can accept a mobile agent

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

(active node) and a device that cannot (passive node). Each passive node must have an associated management station (management node) to be managed. Mobile agents on the network are depicted as a person icon. We also see the composition of a management mobile agent which is mainly its data, its execution state, its intelligence and its abilities. Each management station runs a mobile agent platform and installs basic elements and stationary agents used by mobile agents. Mobile agents can migrate in selected private networks and are not allowed to migrate on public networks unless it is in a strictly controlled manner. This requirement serves the minimal security model explained earlier. Management stations are detailed in Fig. 2. Each management station of the framework runs a mobile agent platform that can receive and launch mobile agents. A station contains a network management mobile agent bank that stores each mobile agent that may be needed to accomplish a task. These mobile agents, through stationary agents, may access local operating system functions and any ability installed on this station.

We begin by describing the technologies employed in the framework and mobile agents. Building on the key framework elements presented earlier, we explain the implementation of management tables, uniform interfaces, stationary agents, and inter-agent communication. Next, we delve into the details of the mobile agents designed to validate the framework. Two mobile agents were developed to detect a set of network failures, without claiming to identify all possible errors. The Diagnostic agent attempts to traverse the network to find the root cause of a failure, a capability that could also be replicated by a stationary agent. The Search agent, on the other hand, is designed to pinpoint the exact cause of a network failure. The framework's implementation utilizes both Java and C++ programming languages, with Java being the primary language used and C++ employed for advanced functions accessed through JNI. This demonstrates the framework's ability to integrate multiple technologies and manage heterogeneous networks. The mobile agents are built using the Grasshopper platform and API (Application Program Interface) (Timon, 2013), leveraging its simplicity, maturity, and support for various operating systems and standards. AdventNet's easy-to-use classes and Java Beans are used to access SNMP elements, with the MIB-II management information base employed."

3. METHODOLOGY

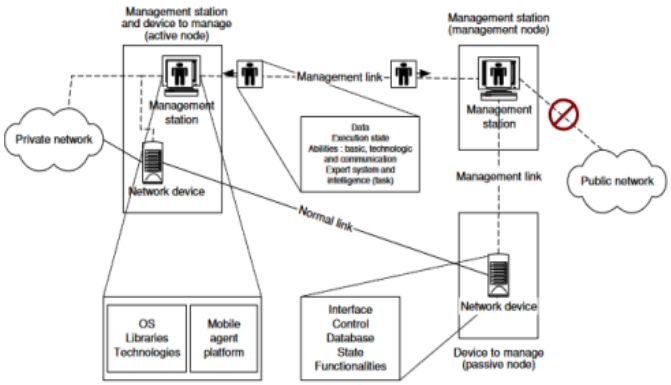
Implementation of the management tables

Management tables are placed on each network management station. The current implementation does not require that a management table be installed on each station, but strongly suggests it. Mobile agents find and access these tables by creating a proxy to the table. We give more information on this type of communication later. The management table implementation uses a hash map to link the management station to network devices. The key of the map is a unique identifier (Table 3), therefore allowing a management station to manage any devices and restrict a device to having only one management station assigned per table.

2.3 Communication, interfaces and stationary agents

In our framework, remote procedure calls (RPCs) serve as the primary communication method between agents, rather than utilizing KQML (Knowledge Query Manipulation Language) or ACL (Agent Communication Language). This approach allows for greater control over communication mechanisms. Additionally, we employ the Grasshopper proxy communication mechanism to facilitate agent communication.

Fig.1: Global View of the Framework



Our framework prioritizes local communication between mobile agents, promoting efficient and localized information exchange.

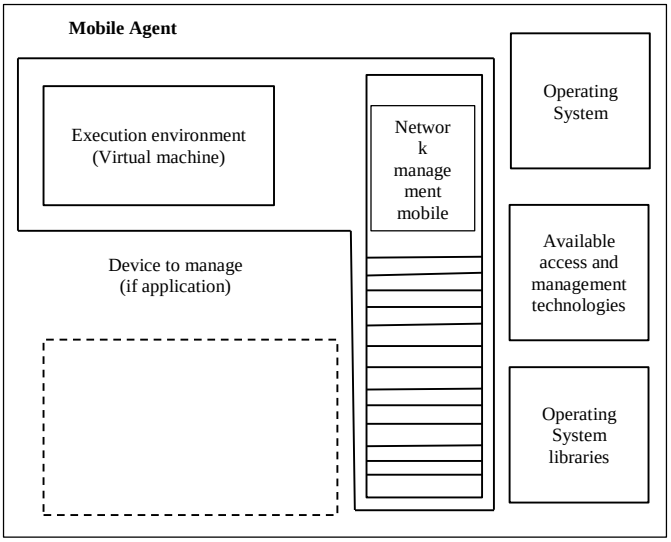


Fig. 2: Detailed view of a management station

and stationary agents on the same place to use as few network resources as possible. Mobile agents that have to manage the network with a global view use proxy communications to access technologies and functionalities that are hidden inside the stationary agents. They therefore lower their need to carry technology dependent code. The framework favours an installation of these agents on each relevant management station. Access to a stationary agent works like a lookup mechanism. For example, let's say that a mobile agent has to manage a device named A. It also knows the interfaces needed for this network management task. The mobile agent then tries to find a stationary agent that implements the interface and has the ability to manage the device A. To do so, it uses a basic set of tests on each stationary agent installed locally. One test allows the mobile agent to test if a stationary agent is able to manage the device. A management station should at least provide one implementation of each interface for each device it has to manage. Otherwise, some devices may not be manageable. Interfaces and functions offered by the framework are given in Table 1. This table is not exhaustive, but is a good snapshot of the abilities of the framework.

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

Table 1: Interfaces and functions of the framework

Parent Ability for all interface	
Functions	Description
Management-station-available	Tell if a component management system is active and may be manage using technology
Base	
Functions	Description
Ping	Ping a network address
Verify-Service	Verify that a given service is available
Performance	
Functions	Description
Get-congested	Return the congestion rate
Get-utilization	Return the utilization rate
Routing	
Functions	Description
Next-component	Return the component physically connected to the device
Next-component	Return the component physically connected to the device to join a given destination
Interrogation	
Functions	Description
Get-Interface-Address	Return all address of an interface
Get-Interface-Information	Return useful information about an interface and state
Get-Interface-Index	Return interface number given an address
Get-Value	Return value of variable or state
Is-Device-Answering	Try to reach the device and return true if is accessible

Table 2: Stationary Agents of the framework and implementation interface

Stationary Mgt. Agent	Implementation Interface
SSnmp	Interrogation Performance Routing
SBase	Base
SWindow	Query Performance Routing

Table 3: Mangement Tables Usede for all Tests

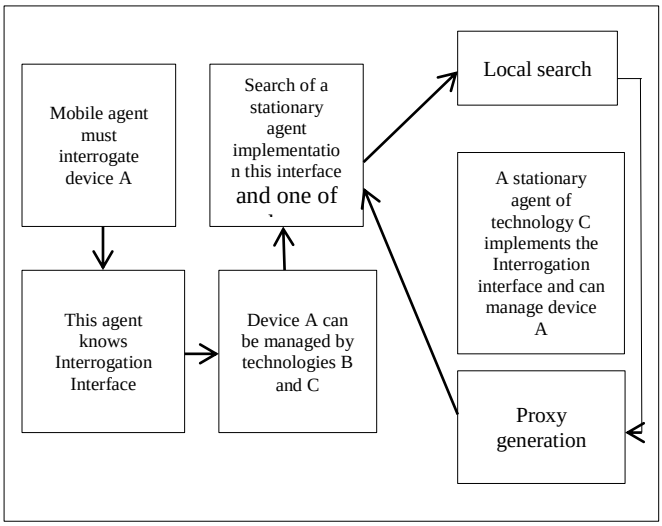
Devices	**Management Stations**
Router Montreal	Montreal
Router Vancouver	Vancouver
Router Boston	Boston
Switch Fidji	Montreal
Management Station Montreal	Montreal
Management Station Vancouver	Vancouver
Management Station Boston	Boston

We used three stationary agents in our framework.

These agents are listed in Table 2. A stationary agent SBase implements server-side functions that are not dependent on the device management technologies. SSnmp is used for network devices and SWindows is used to manage Windows workstations that run mobile agent platforms and are considered as a part of the network to manage.

Diagnostic Agent: One fact is that a network failure could cause many alarms and cause many direct or indirect failures. The diagnostic mobile agent is used when a failure occurs between a source and a destination. It is informed of these two parameters, as well as the port used and nothing more. The diagnostic agent never stops on the first failure. For this reason, its first task is not to diagnose, but accumulate a series of proofs containing facts and places where these proofs are found. Then, at the end of the proof finding phase, it can establish a diagnostic. The proof finding phase ends when the diagnostic agent is unable to move further has moved on or near the destination or has no clues on how to continue (management system down or no route to host). Before terminating, it may try to launch a search agent that returns with an alternate path to the next element. If this agent is slow, a timeout tells the diagnostic agent to continue without waiting longer. The last phase is called the diagnostic phase.

Fig. 4: Diagnostic global Algorithm



http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

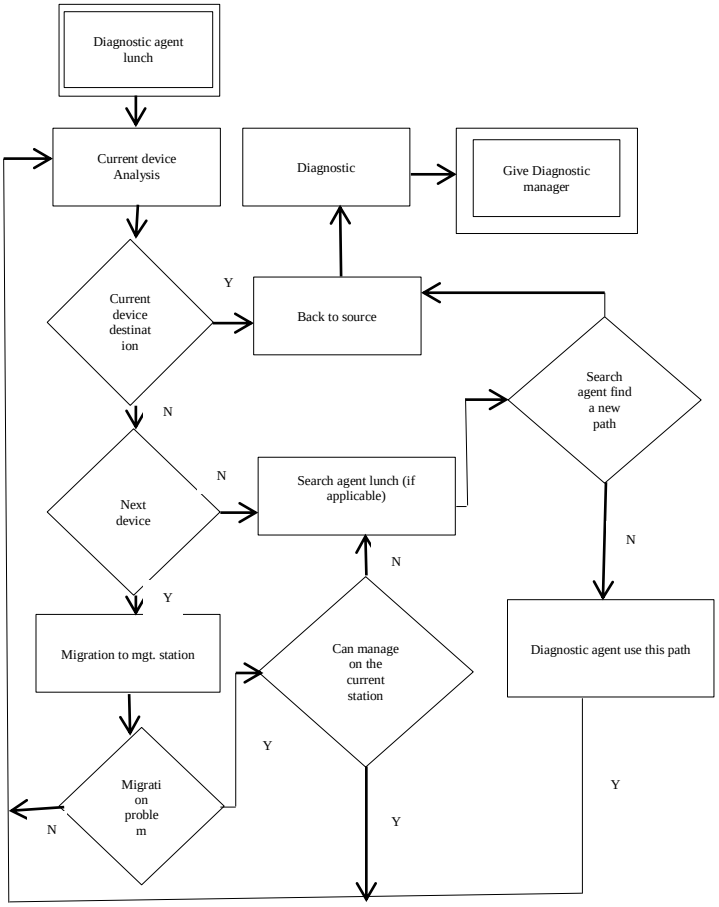


Fig. 5A: Diagnostic Agent Global Algorithm

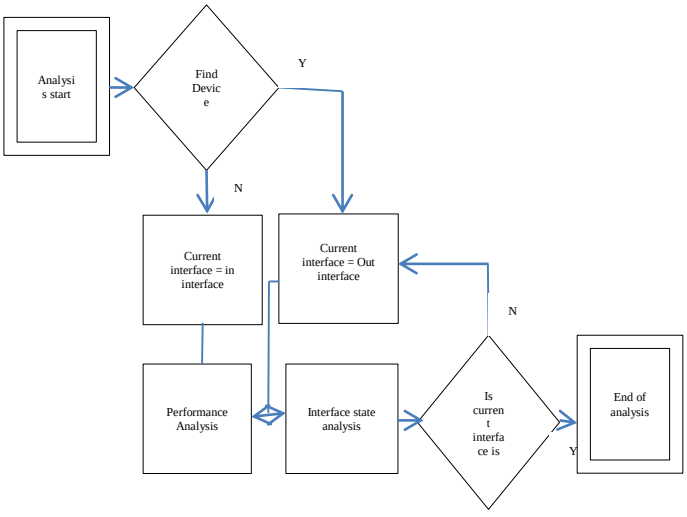
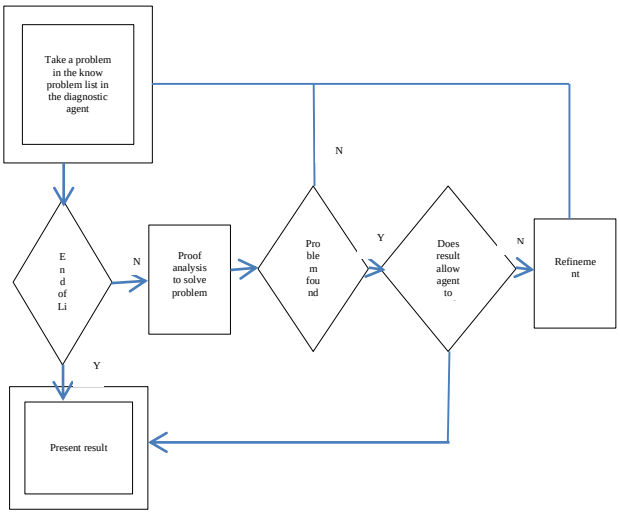


Fig. 5B: Detail Analysis Phase



F

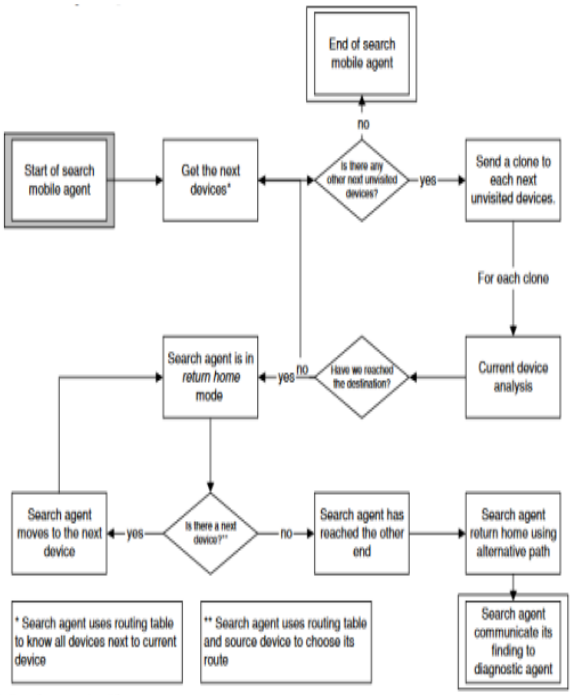


Fig. 7: Search agent algorithm

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

S/n	Management station		Network failure	Failure causes			Solution rating	Response time (s)		
	Source	Destination		Unique	Identified	near		1	2	3
1	Mti	BOS	Link ATM00/1	1	1,2,3		Best	70.34	50.3	76.2
2	Mti	BOS	Link ATM00/0	1	1,2,3		Best	60.34	78.2	70.34
3	Mti	BOS	Interface ATM00/1 on Fidji(Admin down)	1	1,2,3		Best	50.60	34.56	60.34
4	Mti	BOS	Interface ATM00/0 on Fidji (Admin down)	1,2,3	1,2,3		Same	80.45	65.7	50.60
5	Mti	BOS	Interface ATM3/0.1 on Bos (Admin down)	1	1	23	Best	70.2	87.1	80.45
6	Mti	BOS	Link BOS station to hub		1,2,3		Same	60.1	69.3	70.2
7	Mti	BOS	Interface ATM3/0.1 on Mti (Admin down)	1,2,3	1,2,3		Same	23.45	66.7	50.3
8	Mti	VAN	Link VAN to Mti	1	1,2,3		Best	50.3	34.5	78.2
9	Mti	VAN	Interface E0/0 on Mti	1,2,3	1,2,3		Same	78.2	70.1	34.56
10	Mti	VAN	Interface E0/1 on VAN	1	1	23	Best	34.56	67.8	65.7
11	Mti	VAN	Router VAN crashes		1,2,3		Best	65.7	64.2	87.1
12	Mti	VAN	Service down on VAN station		1,2,3		Same	87.1	45.7	65.3
13	Mti	VAN	VAN station crashed	1,2,3	1,2,3		Same	69.3	76.2	50.3
14	Mti	VAN	Link VAN station to VAN router		1,2,3		Same	66.7	70.34	78.2
15	VAN	Mti	Interface E0/0 (Admin down)		1,2,3		Best	87.3	60.34	34.56
16	VAN	Mti	Crashed	1	1		Best	34.5	50.60	65.7
17	VAN	BOS	Interface E0/0 on VAN		1,2,3		Same	70	80.45	87.1
18	VA	BOS	Interface F2/0 on	1,2,	1,2,3		Same	67.	70.	69.

Table 5: Results of test one for all three cases

These phases are detailed in Fig. 4, where a white box inside a gray box indicates the start of the algorithm while a white box inside a white one indicates a possible end. Grayed boxes indicate algorithm portions that are detailed later in Fig. 5 and 6. The proof finding phase starts when the algorithm starts, it ends when the Diagnostic phase is reached and it may be suspended when the mobile agent uses an alternate path given by the search agent. The first thing that the diagnostic agent does is a full analysis of the current device. It then tries to know which device is next on the path between source and destination. By asking the management table, the mobile agent can know which management station is responsible for this device and it tries to migrate on that station. If it is a success, the mobile agent restarts its analysis on the current device and the new management station. If it's not, it tries to manage the device from its current management station. If successful, the algorithm restarts to perform analysis. If not, this tells the diagnostic agent that it has reached a point where it cannot obtain more information. It then has the option of establishing a diagnostic or launching a mobile agent to help find an alternate route (search agent will be explained later). The analysis step is described in Fig. 5. This phase is dependent on which network failures we want to be able to find. The mobile agent does a series of tests without trying to diagnose. A possible optimization here would be to limit the mobile agents to run superfluous tests. For now, this phase has no intelligence. The analysis phase examines a series of facts. These facts are collected in a proof list which is inserted in a path list. The path list is built by collecting information on each interface on the real path between the source and destination. The next detailed phase is the diagnostic phase presented in Fig. 6. The mobile agent intelligence is mostly concentrated in this phase. It tries, using refinement and testing known cause with collected proofs, to know the best location and possible cause that fit current facts about the network. This diagnostic is usually more precise if more facts are found about the problem. It never assumes that the last fact collected is the more relevant for the location or the problem.

Search agent: The search agent clones itself on each route it finds on a given node. Its goal is to find the destination using another path in the network that routing tables may not contain. When it finds the destination, it then tries to come back to the source using routing tables. When it finds a point where it cannot move using these tables, this lets it know that this may be the other end that the diagnostic agent was trying to reach. It then reuses the alternate path to come back to the place where the diagnostic agent is, to give it the extra information. The diagnostic agent then suspends its proof finding phase to move to the element found by the search agent using the alternate path. Arrived at destination, it restarts its proof finding phase. This is a summary of the complete algorithm found in Fig. 7. In case the search agent

http://dx.doi.org/10.46792/fuoyejet.v9i4.2 engineering.fuoye.edu.ng/journal

never returns, the diagnostic agent is still able to give a good estimation of the problem just like a remote management solution. The Search agent is an addition that takes advantage of multi-path networks. To limit its spawn, a hop counter is implemented to terminate itself after too many jumps. This maximum hop value should be set carefully according to network scale and desired precision and performance.

Interactions: To clearly see how the search and diagnostic mobile agents works together, let's look at a brief example illustrated in Fig. 8. In this case, the link between devices A and B is broken. Normally, this will cause one network interface on each of these devices to be automatically deactivated (the operational down state). If the diagnostic mobile agent is used alone, it will see only one deactivated interface. Knowing that the other side is also automatically deactivated will help conclude that something between these two interfaces has gone wrong. If that other interface has been deactivated manually, it becomes apparent that only this interface is the problem. In our case, being able to find an alternate path enables the diagnostic agent to collect more facts about the failure and gives a better diagnostic. This path finding is handled by the search agent. In Fig. 8, the diagnostic agent is stopped at device A. It launches a search agent that finds an alternative route using devices D and E. The search agent comes back to inform the mobile agent of this alternate path. The diagnostic agent may then use this alternate path to pursue its analysis phase on the other end of the failure. It is important to note that on this alternate path, the diagnostic agent does not collect information about devices D and E.

Fine tuning: Our framework leaves mobile agent code on each management station. The mobile agent code is implemented in a class. The real mobile agents that move on the network inherit this class without implementing any new functions. This way, we can tell Grasshopper to only move that lighter inheriting class and install the real code on each management station as core classes. Grasshopper never moves core elements such as its own platform classes and java native classes. What is then moved is only data and execution state and this increases the responsiveness of the system and limits traffic. One drawback is that mobile agent code cannot be updated dynamically. This technique was inspired by the JAMES (Silva, 2019) architecture which uses a more complex system. It uses version checking and only downloads mobile agent's code as needed. Another important technique that we used was to make sure to drop useless data before each movement. This practice is strongly suggested. By useless data, we mean information that is not used anymore, redundant or easy to get at a later time. For instance, after a correlation of alarms, some of those can be typically dropped.

3 RESULTS

Tests: In our preliminary tests, it became apparent that using the diagnostic agent in conjunction with the search agent was an improvement in diagnostic precision, but had two serious side-effects: high response time and high total traffic on the network. We then choose to use two test networks to run our tests. The first one was used to show how easily the combination of the search and the diagnostic agent could locate and diagnose network fault. The easiness was based on the ability of mobile agents to enhance diagnostic precision over stationary mobile agents even if it comes at a high price. This first test network, shown in Fig. 9, offers alternate paths. This test network also has a second goal: evaluate qualitatively the advantages of using management mobile agents in real networks. It unravels areas where mobile agents are better suited than remote management: path finding and searching. This evaluation will be part of our analysis.

On this test network, we made a first test (Test 1) involving twenty random single faults. These twenty faults were simulated for the three following cases: diagnostic and search mobile agents (Case 1), diagnostic mobile agent alone (Case 2) and stationary diagnostic agent alone (Case 3). In each case, we evaluated the precision of the diagnostic and we measured the response times. We also ran a test (Test 2).

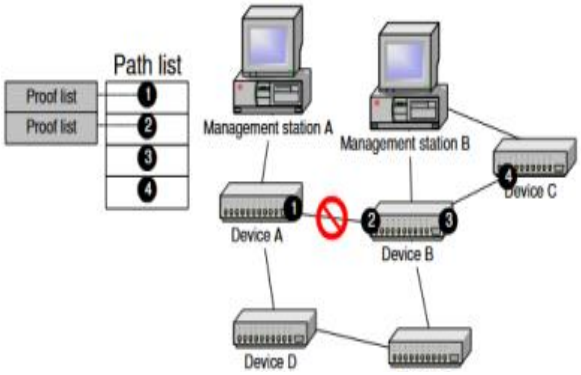


Fig 8: Simple example using search

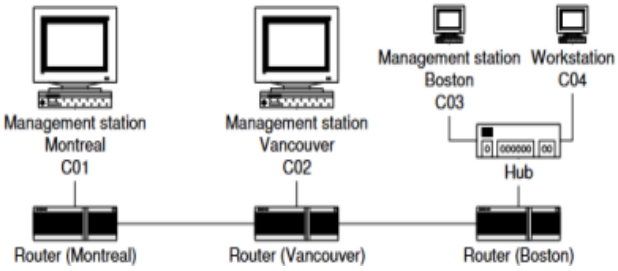
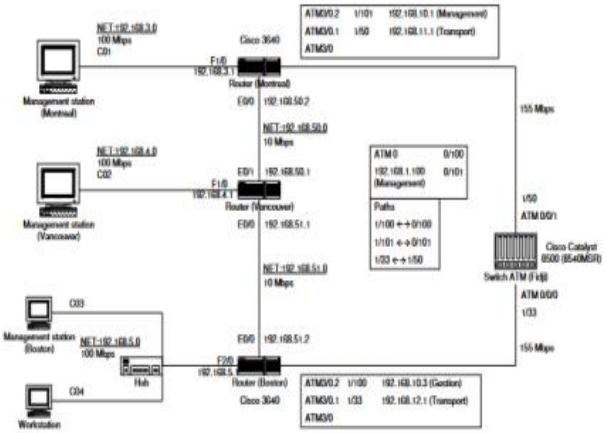


Fig. 9: First test Setup



to demonstrate that the diagnostic agent (mobile or not) was able to know that the management system was in failure. This test is important to be able to discriminate between a network failure involving a loss of service and a management system

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

failure. Another simple test (Test 3) was run to ensure that the diagnostic agent does not mistakenly report network failure in a fully operational network. The second test network's goal was to evaluate mobile agents in terms of raw performance. While this test has already been done for various applications including network management,

Table 6: Special Case experiment (Test 1and test 2)

it was important to know how our framework rates against an equivalent remote approach. To achieve this, we built a test setup that enabled the mobile agent to be the closest possible to the device to manage. It is obvious that being closer to devices should lower the total traffic on the network while load balancing the charge on many devices. What is less obvious is calculating the penalty of moving network management code from one place to another (Gavalas, 2012). The second test network uses a restrained version of the first test network. The result of this subset is a network with only one route from one host to another. The mobile agent can manage each device from the closest management station. This test network makes it easier to test the performance of mobile agents against stationary agents. The last test (Test 4) was limited to failures that imply at least one migration for the mobile diagnostic agent. It does not use the search agent to provide a fair comparison. The mobile agent returns to the source to show its diagnostic, even if it has the ability to do its diagnostic at the destination. The stationary agent uses the same diagnostic algorithm, but is limited to no mobility at all. All tests were made with static routing and single failure scenario. These choices were made to lower the complexity of algorithms and mobile agents. Therefore, mobile agents presented in this study are built for this type of scenario only. It is a limitation for our test, but it still shows possibilities of mobile agents for more typical scenarios. By single failure scenario, we mean that the tests are conducted with only one failure, but this failure may cause more than one alarm and more than one consequence on the network. The content of the management table described is shown in Table 3 and the specifications of the main elements of test networks are shown in Table 4. By reviewing the elements in Table 4, it appears clearly that the focus was on using different types of devices and different transport and management technologies. Our test network can be classified as a heterogeneous network.

4. RESULTS AND DISCUSSIONS

4.1 Results

The results of the first test are shown in Table 5. The results are for the three cases already stated which are: diagnostic and search mobile agents (Case 1), diagnostic mobile agent alone (Case 2) and stationary diagnostic agent alone (Case 3). Here, each table shows the best results for each session in bold. The solution rating column is based on a comparison of Case 1 against the two other cases. It is interesting to note that Case 2 and Case 3 gave the same precision but not the same response times. Session 11 gave the same precision for each case, but we still rate the solution of Case 1 as best because it returned more relevant information about the problem than other cases. The failure cause found is separated in three columns. The first one, denoted unique, indicates that the exact cause of the failure was found and presented as the unique cause. This is the ideal diagnostic for a network administrator. The second one, noted identified, indicates that the cause was identified but lies among a series of other relevant but not exact causes. It may also indicate that the cause was not found precisely, but the right device was found. The last one, called near, indicates that the diagnostic was wrong, but near the cause of the failure. By using the search and diagnostic mobile agents, we always got a better or equivalent precision against the diagnostic agent alone, mobile or not. However, by using the search mobile agent, we significantly increased the response time and the total traffic on the network. The total traffic was not measured

session	Management station source	Dest.	Network failure	Response time (s)	Causes found
1	Mti	BOS	None	12.09	None
2	Mti	VAN	None	7.30	None
3	VAN	Mti	None	12.40	None
4	VAN	BOS	None	6.40	None
5	BOS	Mti	None	10.60	None
6	BOS	VAN	None	8.70	None
7	Mti	BOS	Fidji Mgt. disable	189.70	yes
8	Mti	VAN	Mti Mgt. disable	194.70	yes
9	VAn	BOS	BOS Mgt. disable	195.5	yes
10	BOS	Mti	BOS Mgt. disable	198.7	yes
11	BOS	VAN	VAN Mgt. disable	203.60	yes

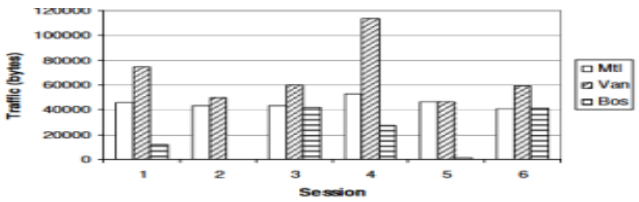


Fig. 11: Traffic measurement for diagnostic mobile agent (test 4)

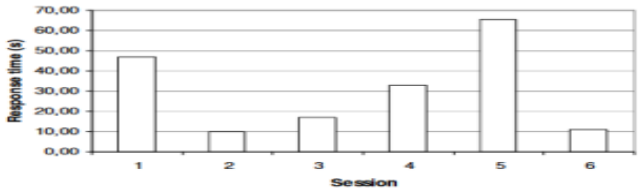


Fig. 12: Response time for diagnostic mobile agent

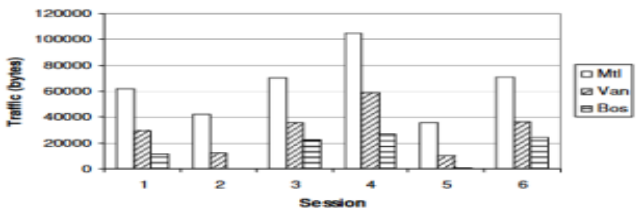


Fig. 13: Traffic measurement for diagnostic stationary agent (test 4)

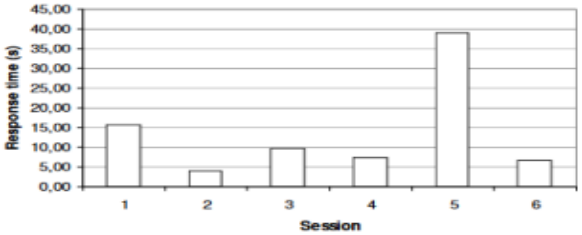


Fig. 14: Response time for diagnostic stationary agent (test 4)

http://dx.doi.org/10.46792/fuoyejet.v9i4.2

for each session but tends to be, on average, eight times greater when the search agent is used.The next two tests are shown in Table 6. As we can see, the diagnostic mobile agent behaved like expected. The diagnostic agent is able to see a difference between a network failure and a management system failure. Also, it behaved as expected in sessions without any network failure. All results we have shown until now on were to evaluate mobile agent technology advantages over remote solutions and stationary agents. The next results give a better idea of the load imposed on networks by mobile agents against stationary agents. The traffic measurements are shown in Fig. 11 and 12 for the diagnostic mobile agent and in Fig. 13 and 14 for the diagnostic stationary agent. Both agents provide the same identification precision. Figures 11 to 14 show that the stationary agent always got a better response time. It also generates less total traffic in each case and less traffic around almost all routers. Fig. 15 shows total traffic value for mobile and stationary agents for each session and Fig. 16 compares each traffic value for each router. To measure traffic values around router, we used out bytes value on each interface of a given router.

4.2 DISCUSSIONS

Initially, the stationary agent appears to be the better choice based on performance. However, we believe that mobile agents will ultimately prove superior, even in terms of performance, when considering the small test network used. Our reasoning is based on the following observations: the total traffic value for complex network tasks is similar for both diagnostic stationary and mobile agents (Fig. 15), and the average network load on the first router is high for the stationary agent (Fig. 16). Complex tasks, such as Sessions 3, 4, and 6, require extensive information querying, which offsets the cost of mobile agent movement. In contrast, Session 5 of Test 4 (Fig. 15) involves minimal information querying, resulting in poorer performance for the diagnostic mobile agent compared to the stationary agent. While the cost of remote management may not justify using mobile agents in small networks, our results suggest that mobile agents will outperform stationary agents in larger networks with multiple simultaneous tasks and higher.

Another point of interest is the network load imposed on equipment near the remote management station. Fig. 16 shows that if we would like to execute more than one management task at the same time, the first router may become a bottleneck. It is now a fact, that mobile agents have a higher response time and may create more traffic overall. However, one main interest about them is their potential of repartition of the network load on the whole network. One bad point about our results is the high traffic variation of 54.8% imposed by mobile agents on the Vancouver router. This is mainly due to the diagnostic agent having to return home and may be optimized further by only sending a report to the source if the result is needed at the source. From here, we did a performance analysis. We are also interested, in the mobile agent domain of research, to find tasks that are enabled only by
mobile agents. So far, it seems that there is no such task. However, what we saw in our experiments are tasks that are more easily executed by mobile agents. For example, we have demonstrated that the search and diagnostic mobile agents were able to find more precisely a cause of a network failure by finding alternate paths to gather more data about the failure (Table 5). In seven of the twenty random failures, they were able to find a better solution. In these seven cases, mobile agents were able to gain access easily to interesting information about the failure that remote management was not able to see. This precision seems to be related to the number of alternate paths a network offers. Also, another point of interest is that diagnostic mobile agents can still work efficiently in an unreliable network whereas its remote counterpart may have a hard time doing the same task. Finally, they offer ways to use existing network management utilities and facilities by their ability to access local resources directly and efficiently.

5. CONCLUSION

This study proposed a network management framework utilizing mobile agents, presented key components of the framework, and discussed implementation details. Experimental results demonstrated the effectiveness of mobile agents in executing network management tasks, particularly search and diagnostic tasks. The framework achieved its two primary objectives: creating a mobile agent-based network management framework for real-world applications and ensuring compatibility with existing networks. However, the framework has several limitations that need to be addressed, including security, fault-tolerance, and limited management capabilities. Future improvements include expanding the set of management functionalities, enhancing the diagnostic mobile agent to handle various network types and errors, and integrating machine learning techniques to improve the diagnostic algorithm. While mobile agents show promise in automating network fault localization and diagnostic tasks, further research is needed to overcome the current limitations and fully realize their potential."

REFERENCES

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal

Baldi, M., S. Gai and G.P. Picco, 1997. Exploiting code mobility in decentralized and flexible network management. Proc. Intl. Workshop on Mobile Agents, MA'97, pp: 13-26.

Baràn, B. and R. Sosa, 2000. A new approach for antnet routing. Proc. Intl. Conf. on Computer Communications and Networks, ICCCN'00, pp: 303-308

Bellavista, P., A. Corradi and C. Stefanelli, 2023. An open secure mobile agent framework for systems management. J. Network and System Management, 7: 323-339.

Bieszczad, A., B. Pagurek and T. White, 1998. Mobile agents for network management. IEEE Communications Surveys, 1: 1.

Bohoris, C., G. Pavlou and A. Liotta, 2023. Mobile agent-based performance management for the virtual home environment. J. Network and System Management, 11: 133-149.

Borselius, N., 2002. Mobile agent security. Electronics & Communication Engineering J., 14: 211-218.

Bossardt, M., L. Ruf, B. Plattner and R. Stadler, 2000. Service deployment on high performance works nodes. Proc. IEEE/IFIP Network Operations and Management Symposium, NOMS'02, pp: 915-917.

Boyer, J., B. Pagurek and T. White, 1999. Methodologies for PVC configuration in heterogeneous ATM. pp: 211-228.

Di Caro, G. and M. Dorigo, 1998. Ant colonies for adaptive routing in packet-switched communications networks. Proc. Parallel Problem Solving from Nature, PPSN'98, pp: 673-682.

Gavalas, D., D. Greenwood, M. Ghanbari and M. O'Mahony, 2002. Hierarchical network management: A scalable and dynamic mobile agent-based approach. Computer Networks, 38: 693-711.

Gavalas, D., D. Greenwood, M. Ghanbari and M. O'Mahony, 1999. An infrastructure for distributed and dynamic network management based on mobile agent technology. Proc. IEEE Intl. Conf. on Communications, ICC'99, pp: 1362-1366.

Liotta, A., G. Pavlou and G. Knight, 2022. Exploiting agent mobility for large-scale network monitoring. IEEE Networks, 16: 7-15.J. Computer Sci., 2 (8): 646-659, 2006 659

Koon-Seng, L. and R. Stadler, 2021. Weaver: Realizing a scalable management paradigm on commodity routers. Proc. IFIP/IEEE Intl. Symp. On Integrated Network Management, IM'03, pp: 409-424.

Pagurek, B., Y. Wang and T. White, 2000. Integration of mobile agents with SNMP: Why and how. Proc. IEEE/IFIP Network Operations and Management Symposium, NOMS'00, pp: 609-622.

Puliafito, A. and O. Tomarchio, 1999. Advanced network management functionalities through the use of mobile software agents. Proc. Intl. Workshop on Intelligent Agents for Telecommunications Applications, IATA'99, Springer-Verlag, pp: 33-45.

Putzolu, D., S. Bakshi, S. Yadav and R. Yavatkar, 2000. The phoenix framework: A practical architecture for programmable networks. IEEE Commun. Mag., 38: 160-165.

Rayan Stephan, Pradeep Ray, N. Paramesh, 2023. Network management platform based on mobile agents. An international journal of Network Management, 10: 1002.

Rossier, D. and R. Scheurer, 2020. An ecosystem-inspired mobile agent middleware for active network management. Proc. Intl. Workshop on Mobile Agents for Telecommunication Applications, MATA'02, pp: 73-82.

Rubinstein, M.G., O. Duarte and G. Pujolle, 2000. Reducing the response time in network management by using multiple mobile agents. Proc. Third Intl. Conf. on Management of Multimedia Networks and Services, Kluwer Academic Publishers, pp: 253-265.

Rubinstein, M.G., O. Duarte and G. Pujolle, 2002. Scalability of a network management application based on mobile agents. J. Communications and Networks, 5: 240-248.

Schoonderwoerd, R., O. Holland and J. Bruten, 1997. Ant-Like agent for load balancing in telecommunication networks. Proc. Intl. Conf. on Autonomous Agents, Agent'97, pp: 209-216.

Papavassiliou S., A. Puliafito, O. Tomarchio and J. Ye, 2022. "Mobile agent-based approach for efficient network management and resource allocation: framework and applications," in *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 858-872, doi: 10.1109/JSAC.2022.1003050.

Silva, L.M., P. Simoes, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida and N. Stohr, 2021. JAMES: A platform of mobile agents for the management of telecommunication networks. Proc. Intl. Workshop on Intelligent Agents for Telecommunications Applications, IATA'99, Springer-Verlag, pp: 77-95.

Timon, C. Du, Eldon Y. Li and An-Pin Chang, 2003. Mobile agents in distributed network management. Communications of the ACM, 46: 127-137.

White, T., B. Pagurek and A. Bieszczad, 1999. Network modeling for management applications using intelligent mobile agents. J. Network and Systems Management, 7: 295-321.

White, T. and B. Pagurek, 1998. Towards multi-swarm problem solving in networks. <http://dsp.jpl.nasa.gov/members/payman/swarm/white98-icmas.pdf>Proc. Intl. Conf. on Multi-Agent Systems, ICMAS '98, pp: 333-340.

Zapf, M., K. Herrmann and K. Geihs, 1999. Decentralized SNMP management with mobile agents. Proc. of the Sixth IFIP/IEEE Intl. Symp. On Distributed Management for the Networked Millennium, pp: 623-635.

http://dx.doi.org/10.46792/fuoyejet.v9i4.2
engineering.fuoye.edu.ng/journal