Exploiting the Usage of Mobile Agents as a Network Performance Monitoring Tool for Network Fault Management – an Alternative to Other Traditional Approaches

Isaac Bansah
Information Technology Department, Methodist University College, Accra, Ghana

Tonny Montana Adegboyega
Information Technology Department, Methodist University College, Accra, Ghana

ABSTRACT
The growth of heterogeneous networks has increased the demand of adopting robust and an effective technology that can cope with the trend. Developers have always thought of an effective way to manage a network taking in to consideration the current growth trend and the complexities of modern network architecture. Extensive research yielded a concept called the Mobile Agent paradigm. It came as an alternative to the traditional client server paradigm has since proven to be the obvious choice as far as effective network management is concerned
This paper addresses the exploitation of Mobile Agents as a Network Performance monitoring tool. The mobile agent agents used is a prototype owned by IBM Tokyo called Aglet. The aim is to provide an intelligent codebase in java for the Aglet to retrieve performance variables and other relevant data on a virtual network. Results are analyzed and conclusions are drawn based on the analysis and evaluation

General terms
Client-Server Paradigm, Remote Procedure Call, Mobile Agent Migration

Key words
Network Management, Fault Management, Performance Management, Mobile Agents, Java, Aglet

1. INTRODUCTION
Mobile agent technology has been accepted as an alternative solution for the challenges posed by managing large and complicated network architectures. Its mobility and communication features are believed to surmount over all other paradigms. It shares similarities with java applets but what makes it superior is the fact that apart from it carrying its class along, it is also capable of carrying its executable state across the network. It is also capable of executing its state on any host on the network.
The prototype developed by IBM Tokyo called Aglet is the one to be used for this task. The Aglet codebase is written in Java which makes the aglet inherit all benefits and policies java provides as well as the Aglet policy.
The object-oriented form of programming provided in java helps to define appropriate variables in the codebase.
Mobile agents have proven to be the appropriate technology for this task due to its robust and flexible nature.

2. CLIENT/SERVER PARADIGM
The Client/Server paradigm is a form of computer network paradigm that involves request and dispatch of information between the client and the server and has related remote procedure call mechanism as follows: [1]

Client
- Any application program
- Makes a request
- Awaits a response

Server
- Specialized program (process)
- Awaits a request
- Computes an answer
- Issues a response

[2] Cited [3] where the remote procedure call (RPC) is an interaction exercise to remedy these shortcomings by allow client request transactions to be performed in the same way as a local function call is made; all stages involved must be transparent to the client
[4] also described the Client –Server paradigm as the traditional design approach for distributed communication among sites, in which messages are transferred from one site
to another, but actual code is not.
[5] registered a concern about the fact that synchronization points and network addresses would always have to be determined by developers because of the fact that the Client-Server paradigm is too much of a low-level one. It went on to say the exact services provided by the client must be known as the Client-Server interaction is too specific.

The initial contact is always by the client to server in form of information or service request. The server in this case has all the resources and based on the kind of resource requested by the client, the server honors it and execute as the client has not got the resources to do so. [6] described the client as not intelligent enough to execute this requests since the server has all the know-how, processors and resources.
[7] Cited [8] who described the Message Passing and Remote-Procedure calls as a very reliable means of client-server communication in distributed applications.
These highlights the limitations this paradigm offers when put to use though it is still supported by a couple of technologies.

2.1 Code on Demand Paradigm

This form of paradigm is often described as a mobile code and its characteristics are typically exhibited by java applets. Unlike the client server paradigm, the client has some resources and the ability to process information.

The client makes request to a host for codes to execute specific tasks locally. The codes are then sent to the client by the host on the network.

[9] Cited [10] in his article describing the server as a code repository whiles changing behavior of the client upon successful execution of the codes it supplies.

[11] made mention of the Remote Evaluation Architecture, where in this case the server is required to execute a code that is sent from a requesting client and returns the results to the clients; a direct reversal of the code in demand situation.

2.2 Mobile Agents

Mobile agents have been defined in so many ways by different writers. [12], defined Mobile Agents as a software entity which exists in a software environment. [13] also defined mobile agents as an agent that is capable of moving between machines. It was also described by [6] as an agent that has the unique ability of transporting itself from one network to the other. In short mobile agent is a form of software agent that is capable of traveling across a network.

[6] mentioned that agents are capable of transporting itself with its state and code. The state of the mobile agents consists of the attributes and values of the mobile agent that helps in its executions of its functions and the code is the class code of the mobile agent in terms of mobile orientation.

The mobile agent needs to maintain the same state and code while traveling down a network for unique identification and form for execution of its functions at any point of the network. The mobile agent is only capable of executing its functions or interacts with an object across a network with the same host. One feature of a mobile agent is the fact that it is heterogeneous which makes it capable of traveling across heterogeneous.

Mobile agents have been accepted, tested and proven to be the most effective way of managing networks. It allows flexibility, automation and security. [14]

3. NETWORK MANAGEMENT OVERVIEW

As computer networks grow, the task of managing also grows. Large and complicated networks demanded effective management practices to ensure optimal and efficient performance. Adapting an efficient network management system to keep with the growth is a need. [15] described network management as a list of activities performed on a network to ensure smooth and efficient running with minimal downtime. The amount of downtime experienced by a network determines the reliability of the network.

3.1 OSI Management Functions

Managing all aspects communication network requires many tools. [16] Open System Interconnection (OSI) management functions are required to be met by the activities of managing a network. To successfully manage a network requires all these functions to be carried out effectively. Listed below are the five OSI management function practices: [17].

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

3.2 Fault Management

Fault management as one of the OSI management functions captures activities like error detection, tracing and identifying faults, information logging and fault ticketing as some of the practices undertaken when managing faults. [18]. Other activities involved in fault management includes executing series of diagnostics tests, correcting faults as well as reporting error conditions. Others including thorough and careful examination of available information to help localize and trace fault effectively. [18]

As networks keeps growing and its nature looking more complicated, huge and heterogeneous, so is the need and desire for network administrators to employ prudent and sustainable fault management practices to maintain and cope with the pace of growth. This trend has automatically rendered human administrators obsolete as automated fault management systems are being adopted for efficiency. The fault domain of a network increases as the network grow larger and complicated because the hardware and the software requirements of the physical network also increases. Tracking and detection, logging and ticketing, isolation and resolution , to mention a few, are also some of the functions of fault management. [17]. Using the above procedures would initially involve identifying faults by possibly comparing sequences and later observing previously logged faults patterns or better still identifying the fault uniquely. A suitable solution is proposed after the exercise and records or procedure of execution is recorded for future references. Fault management according to [19] involves these five step processes

- Fault detection
- Fault location
- Service restoration
- Identification of problem root cause
- Problem resolution

These two authors seem to be addressing the same problem with similar processes but varying steps. Both have emphasized the need for a systematic approach to fault management. These approaches allow dynamism and give the opportunity for feedback or revision of any of the steps involved for re-examination. Each of the stages involved is very critical as the next stage can only start at the end of the previous. Diagnostic tools, network applications and applications and platforms have subsequently being developed to make fault management simple and effective. This allows the flexibility of accessing and managing fault remotely. [20] mentioned that damages created by fault and its cost of repair should be monitored by collecting data on the effectiveness of the fault management process. These data can then be normalized to account for network specifications and parameters for which can be intend be used to determine the performance of the fault management process. Good monitoring and controlling practices keeps track of the management and alleviate any possibility of deviation, mistakes or bad practices. Ensuring an optimum level of performance of the fault management process guarantees efficiency and effectiveness of the whole process. [21]

4. AGLET OVERVIEW

In the early 1995s, the research laboratory of IBM Tokyo developed the Aglet API as an agent development kit. [6]. Danny B Lange, a member of the research team initiated this effort as cited by [6].

Aglets can simply be defined as java mobile agents. Aglets as java objects are capable of transporting themselves from one host to the other on a network while maintaining its code and data.
The earlier versions of Aglets developed was version 1.0.3 which later through the open source project became available including the later versions (Aglets 2.0.2). [6] also mentioned that you can think of an Aglet being generalized and extended for java applets and Servlets. It also acknowledges the concept of having its itinerary dynamically routed and executing autonomously.
Aglets execute in an Aglet server host. The default Aglet server host is called Tahiti.

4.1 Requirements

Mobile agent technology is now being adopted by both developers and researchers to help solve some of the lapses created by other traditional methodologies.
The ability of using mobile agents in various forms of network solutions has been proven by good extensive work by researchers and developers. This concept clearly demonstrated superior performance which made it a preferred choice over the traditional sever-client paradigm
The aim of this project is to use aglet (mobile agent) to remotely monitor network performance for fault management purposes. The monitoring would be done on a virtual network with different subnets.

4.1.1 Requirement Analysis

Writing a codebase for aglets or developing an application using agents gives the developer the opportunity to investigate and find data of a particular area of concerned. Extensive research has been done in various forms for the use of mobile agents for network monitoring purposes. Most of these research activities were based on the potential and basic functionality of the mobile agent but not associating it with spe4cific tasks for that matter putting it to test for specific duties. Developing an application to retrieve network statistics like, throughput, delay, availability and retrieving event log files remotely demonstrates the practicality and the real situation benefits of using mobile agents for monitoring purposes.
Retrieving these network statistics could briefly give one an idea about the state of the network. The throughput and the delay might also be tested against the varying load size of aglet.
Below is the listed summary of requirements:

- To be able to analyze data collected to tell the state of the network
- To be able to use information retrieved to manage network resource allocation and usage
- To be able to remotely monitor a network by diagnosing, repairing and retrieving relevant network data for network fault management.

4.2 Aglet Design Requirement

Aglet requires Aglet Software Development Kit (ASDK) which is an open source program by IBM Tokyo to develop an aglet. It is available for download on the internet for the earlier version of 1.1.3 or later which in this case is 2.5.0. It requires an additional software environment of a JDK platform which in this case as J2SDK version 1.4.5_09. It is also an open source program own by sun Microsystems and available for download on the internet.
The ASDK package comes with a default server called Tahiti which provides a Graphic User interface (GUI) for manipulation of the aglet.

4.3 Aglet Data Retrieval

Quality or level of data retrieved bay an aglet is partly determined by the intelligent request in the codebase. Data retrieved by aglet is normally stored in an array. Effective use of an array is required as an overload or storing too much information on a single array may affect the performance of the memory requirements.
It is always recommended to store each array in a single vector for optimal performance. Vectors with singe stored arrays are capable of expanding to accommodate more arrays as and when needed.

5. AGLETS PROTOTYPE

The first issue considered in designing the aglets prototype was the aglet being able to serve its purpose by retrieving the required data. The accuracy of the data was not much concern at this stage as all the remote hosts on the network were using copies of the resources and processors available on the host machine. The main aim was to demonstrate the possibility of the aglet the task its being designed for.
Security was a major problem which prevented data retrieval remotely. Further research revealed how permissions could be enabled to allow read and write permissions to aglets.
The first move was to disable the aglets.props file in the cnf directory of the aglet.
It is done by setting it to false which disables the aglet security as shown below:
Aglet.secure=true – original state
Aglet.secure=false- disabled state

Before using the OnCreation method to create the aglet, the vectors are defined and variables stated in the extended class of the aglet.
The variables defined included the home variable which is meant to record the dispatch location of the created aglet and also serves as a ‘reminder’ for where the aglet finally returns. This is done by obtaining the aglet context and the URL of the host as shown below:

```
Home = getAgletContext().getHostingURL().toString();
AgletContext ac = getAgletContext();
```

Specifying the travel itinerary was next as it needs to know where it’s going. The itinerary is normally listed depending on the number of destinations to visit and what pattern to follow. The travel pattern of this aglet was to start from the first URL address through to the last one ad return. Another option is to dispatch randomly or broadcast. Shown below is the travel itinerary of the aglet.

```
itinerary.addElement (new  URL (“atp://Isaac:7201”));
itinerary.addElement (new  URL (“atp://Isaac:7402”));
itinerary.addElement (new  URL (“atp://VM1:7601”));
itinerary.addElement (new  URL (“atp://VM1:7804”));
itinerary.addElement (new  URL (“atp://nana:8105”));
itinerary.addElement (new  URL (“atp://nana:8306”));
```

The next stage involved deciding on the aglets packets to be transmitted across the network for the next. It was decided to measure the throughput and latency against five aglets packet and also to test the effect of load on these performance variables. This was done by allowing the original aglets to clone itself five times and dispatch them automatically.
This was done by introducing a new ‘isClone’ Boolean in the extended aglet class and was set to false. Anew private vector and string was also defined for cloned copies.

```
Boolean isClone = false;
Private Vector nanas;
Private String nanaName = null;
```

The five copies were cloned and time interval was indicated to space up the cloning process. This allows the dispatch time of each cloned aglet to be note. Created aglets are automatically dispatched since the itineraries are specified in the code. Each of the cloned aglets carries that same code as the original. The cloning and the specified time for one copy are shown below:

```
Nanas = new vector();
        {
        Try {
                Thread.sleep(1000);
                } catch
(InterruptedException e) {}
        }
```

Subsequent copies were made with varying time intervals. A clone listener and adapter were added to allow the use of the OnClone Method. This method is used to change the value of the Boolean variable 'isclone' to false which prevents the clone from cloning itself again.

```
addCloneListener(new CloneAdapter() {
        public void onClone(CloneEvent
ce){
                isClone = true;
        }
});
```

During the cloning process, the original aglet was delayed to accommodate to time spent in cloning and dispatching.
The throughput was obtained taking in to consideration the total aglet packets sents, the time interval between their dispatches and the latency attained by the last aglet packet to arrive.
The latency was also calculated by the time difference between the start time of the first aglet and finish time of the last aglet.
Since all the aglets including the clones carries the same codebase, the last aglet to arrive returns the required value of the throughput and latency. Accordingly, the diagnosing time was also obtained by reading the time the aglet reached the host and the time it leaves the host.

```
long latency = finish-start;
long latencyInSec = latency;

//the thread sleeptime is converted to seconds.
//and the latency is also converted from milliseconds.
//to seconds
long throughput =
5/1+2+2=3+2+(latencyInSec/1000);

long delayInServer = finishserver-startserver;
long delayinServerInSec = delayInServer;
system.out.println("\nDisagnosis time: " +
delayinServerInSec + "milliseconds.\n");
system.out.println("\nNetwork Latency: " +
latencyInSec + "milliseconds.\n");
system.out.println("\nDisagnosis throughput: " +
throughput + "bytes/seconds.\n");
```

The calculation of the throughput was on this formula deduced from the pattern of dispatch.

Throughput = n/T1+T2+T3+T4+……Ln

Where;

n = number of aglet packets
T1….Tn = time of departure of aglets
Ln = Latency of last aglet to arrive

A vector of netInfoData is added for storage of retrieved data which is eventually displayed or written to file.
For the network to be tested against varied aglet packet sizes (load), a public string was introduced that allow the file size to be increased as shown below:

```
Public static Sting PacketSize=
"abcdefghijklmnopqrstuvwxyz,
abcdefghijklmnopqrstuvwxyz"+
"abcdefghijklmnopqrstuvwxyz,
abcdefghijklmnopqrstuvwxyz";
```

The aglets, after dispatch, travels to its specified destinations. Aglets packet size was varied to evaluate the effect of load on the network throughput and latency. There were two outs to data retrieved from this test as described below:

- Username
- PC Architecture
- Operating system
- Operating system Version
- Architecture
- Computer name
- URL of java vendor

The first group of data above was printed to command prompt based on the assumption that it was meaningful at its state without any further analysis.
The second group of data below was written to file for analysis purposes.

- Throughput
- Latency
- Diagnosing Time

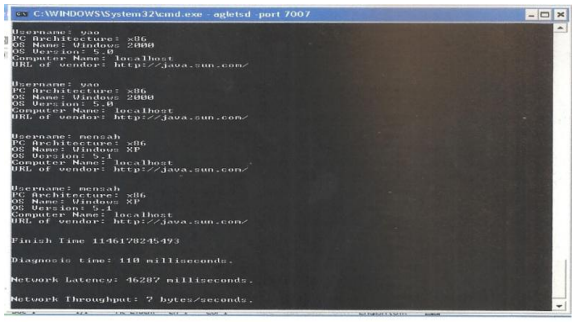

Figure 1. Preview of report printed to command prompt

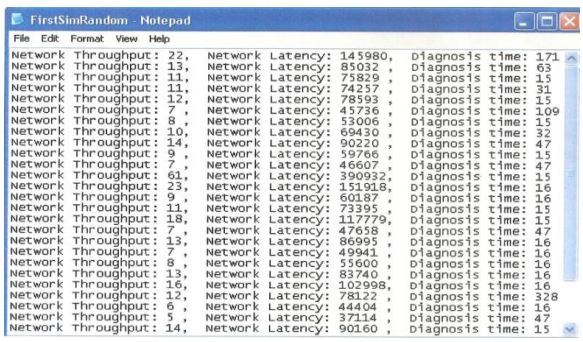

Figure 2. Preview of data written to file

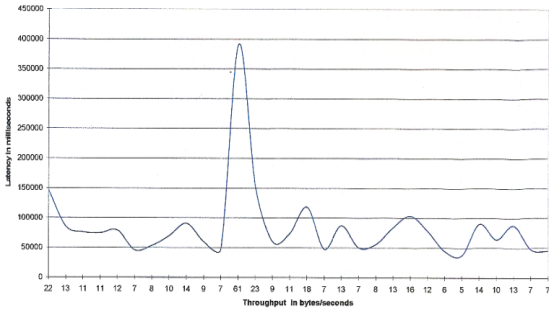

Figure 3. Chart of latency against throughput with 11.35KB packet size of Aglets

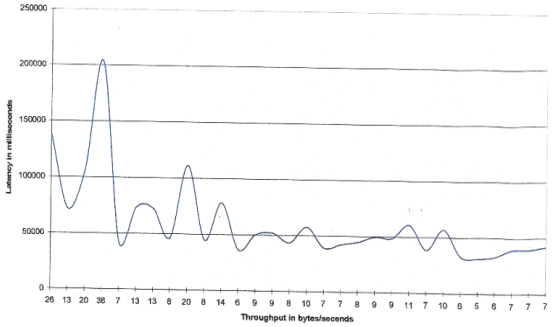

Figure 4. Chart of latency against throughput with 26.5KB packet size of Aglets

6. EFFECTS OF LOAD ON THROUGHPUT AND LATENCY

Referencing the chart developed from the throughput and latency figures obtained, the highest throughput value registered the highest latency of the test where the lowest throughput value registered the lowest latency.

This is due to the fact that at the highest throughput state, there might have been congestion on the available bandwidth in an attempt to send that size of packet across the network. This might have resulted in retransmission of unsent packets which intends courses accumulation of latency.

The smaller packets had low latency due to availability of bandwidth hence the transfer rate was not impeded in anyway apart from the default delay that the system might provide.

The system was tested against different sizes of aglet packets. The behavior pattern of the throughput and latencies were similar. But in the case of the higher load, maximum latency attained at its highest throughput was higher than the maximum attained in the case of lesser aglet size. Similarly the minimum latency obtained by the smallest throughput value was also higher than the one obtained in the smaller aglet packet size as shown in the previous charts. This shows that the network clearly reacted to the change in packet sizes.

This pattern clearly shows that the size of the load could be directly proportional to the latency having the Bandwidth as a constant.

7. CONCLUSION

Mobile agent, a computer network multi-functional tool has proven to be the technology of the future. Numerous exploitation and excessive and experimental work have testified the success story of this concept. The mobile concept was born due to ever growing nature of computer networks. The complexity of designs and architecture needed a more robust and efficient way of managing it.

Mobile agents were tried, tested and adopted to be the appropriate technology for this challenge. This concept was chosen over old paradigms like code on demand and the client and server. This was the motivation behind this publication. Putting the mobile concept to test again by using it as performance monitoring tool was a step forward in stretching the capabilities of the mobile agent and confirming its characteristic of been flexible.

Aglet which is mobile agent with a java code base offers the same services as any other mobile agent. The intelligence of an aglet depends on what the developer wants to achieve. The accuracy of the data retrieved might also depend on the assumptions made by the developer. Assumptions might be made taking in to consideration previous fault patterns, logs and tickets.

Throughput and latency are major determinants on a computer network as far as performance is concerned. The ultimate aim of managing any network is to ensure optimal performance variables at an acceptable level.

What this paper addressed was exploiting the possibility of using mobile agents to monitor these performance variables to aid effective fault management. Effective fault management practices reduce downtime of networks and improve network availability.

There is nothing important and appropriate than a network administrator being able to manage a network remotely. Apart from that, it saves time, cost and labor, the convenience it provides is unquestionable.

8. REFERENCE

[1] Ostermann S., Comer D., 2011 Networking Applications Available from: http://oucsace.cs.ohiou.edu/~osterman/class/cs544.archive/notes/apps.pdf

[2] Reinhartz-Berger I., Dori D. and Katz S., 2005 'Modelling code mobility and migration: an OPM/Web approach', Int. J. Web Engineering and Technology, Vol. 2, No. 1, pp.6–28.

[3] Bloomer, J. 1992 Power Programming with RPC, Sebastopol, CA: O'Reilly and Associates.

[4] Renaud, P.E. 1993 Introduction to Client/Server Systems: A Practical Guide for Systems Professionals, Hoboken, NJ: Wiley & Sons.

[5] Dale, J. and DeRoure, D. 1997 'A mobile agent architecture to support distributed resource information management', Proceedings of the International Workshop on the Virtual Multicomputer. Available from: http://www.mmrg.ecs.soton.ac.uk/publications/archive/dale1997b/vim97.pdf

[6] Lange D.B., and Oshima M., 1998. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley longman, Inc.

[7] Al-Kasassbeh M, Adda M.(2007) The analysis of mobile agent in network fault management.

[8] Nikaein N. Reactive autonomous mobile agent, 1999.

[9] Picco G. P., 2000. Glossary of Code Mobility Terms. Available from: http://mucode.sourcefourge.net/docs/info/glossary.html

[10] Fuggetta, G. P. Picco, and G. Vigna Understanding Code Mobility IEEE Transactions on Software Engineering archive Vol. 24, No. 5, May 1998.

[11] Bohoris C. Network performance management using mobile software agents. In: School of electronic engineering, information technology and mathematics, vol. Doctor of Philosophy Guildford: University of Surrey, June 2003. p. 154.

[12] Chess D., Grosof B., Harrison C., Levine D., 1995 Itinerant agents for mobile computing. IEEE Personal Commun 1995;2:34.

[13] Tanenbaum A. S., 1996 Computer Networks. London: Prentice Hall

[14] Hovart D., Cvetkovic D., Milutinovic V., Kocovic P., and Kovacevic V., 2000. Mobile Agents and Java Mobile Agents Toolkits, Proceedings of the 33rd HICSS, IEEE

[15] Carr J., 1990 Network Management. [Online]. Available from http://www.itarchitect.com/article/NMG20000724S0049

[16] Bieszczad A., Pagurek B., And White T., (1998) Mobile Agents For Network Management IEEE Communications Surveys • http://www.comsoc.org/pubs/surveys • Fourth Quarter 1998 • Vol. 1 No. 1

[17] Held G., 1992. Network Management – Techniques, Tools and Systems. John Wiley & Sons, Ltd., West Sussex England.

[18] Anonymous, 2001. Fault Management

[19] Subramanian M., 2000 Network Management – Principles and Practice. Addison Wesley Longman, Inc.

[20] Byrne C. J., 1994. Fault Management – Telecommunications Network Management in the 21st Century. New York: IEEE Press.

[21] Terplan K., 1996. Effective Management of local Area Network. 2nd Ed. The McGraw Hill Companies Inc