Available online at www.sciencedirect.com
ScienceDirect
Journal of Network and Computer Applications 31 (2008) 699–711
Journal of NETWORK and COMPUTER APPLICATIONS
www.elsevier.com/locate/jnca

Analysis of mobile agents in network fault management

Mouhammd Al-Kasassbeh*, Mo Adda

School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK

Received 18 January 2007; received in revised form 27 September 2007; accepted 1 November 2007

Abstract

Network domains have become more and more advanced in terms of their size, complexity and the level of heterogeneity. Comprehensive fault management is the most significant challenge in network management. Fault management can help increase the availability of the network by quickly identifying the faults and then, proactively, start the recovery process. Current centralized configured network management systems suffer from problems such as insufficient scalability, availability and flexibility as networks become more distributed. Mobile agents (MAs), with integral intelligence, can present a reasonable new technology that will help to achieve distributed management, several researchers have embraced these approaches. In this paper, we introduce a general analytical model for network management client/server (CS) and MA paradigms. We express how to build up an analytical framework, which can be used to quantitatively assess the performances of the MA and CS paradigms under different scenarios. We present some numerical and experimental results that demonstrate the applicability of our proposed framework, which will be based on a combination of MA and CS schemes called Adaptive Intelligent Mobile Agent.

Keywords: Mobile agent; Distributed network management; Software agent; Network fault management

1. Introduction

At present, the most popular model for distributed computing is the client–server (CS) paradigm. Conversely, with rising demands on processing power and the need to conserve

*Corresponding author. Tel.: +44 7884 226298.
E-mail addresses: mouhammd.al-kasassbeh@port.ac.uk (M. Al-Kasassbeh), mo.adda@port.ac.uk (M. Adda).

1084-8045/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2007.11.005

bandwidth on large and slow networks, the restrictions of this approach have become apparent. The CS model is suitable for the applications with a limited need for distributed control (Pleisch and Schiper, 2004). Undoubtedly, this limits the flexibility with which a client can use a server by processing the server data locally according to the client's needs. Thus, the mobile code-computing paradigm was introduced. With this paradigm, the data and the code are transmitted together between the client and the server. Transmitting the code to the destinations helps in terms of scalability and adaptability.

Primarily, distributed applications rely on the CS paradigm, in which the client and the server communicate through message-passing (MP) or remote-procedure calls (RPC) (Nikaein, 1999). RPC is usually synchronous; the client suspends itself after sending a request to the server, waiting for the results of the call. A different approach is called code on demand (COD) (Carzaniga et al., 1997). In COD, a code object is sent from the server to the client, the client downloads the code in order to get dynamic interaction. Another architecture is called remote evaluation (REV) (Bohoris, 2003). In REV, the client sends its own code to a server, requesting that server to execute the code in order to return the results.

More recently the concept of a mobile agent (MA) has been introduced. A MA could migrate between nodes, carrying the necessary code to be executed (Nikaein, 1999). MAs offer a real benefit to the system developers and the users; this application specifying which parameters can be improved upon, for example performance, connectivity, reliability and modularity. The importance of these key parameters may vary from one application to another. In this paper, we derive analytical models, and support them by experiment results for CS and MA paradigms. We will deduce that under certain conditions such as the size of the network and the size of the agents, CS has better performance than MA.

2. What are mobile agents?

According to Chess et al. (1995), a MA is a software entity which exists in a software environment. So far, there is not a common definition for MAs. The problem of arriving at an agreed definition of what constitutes an agent is down to identifying the features that distinguish an agent from a common computational entity. This has raised controversial arguments for nearly a decade and only recently have some features been identified (Liotta, 2001). The agents used focus on software systems and, consequently, the term agent will be defined and used for software systems only, i.e. each agent is a software agent and thus an independent program. However, even with this restriction, typical definitions of agents are not precise (Richard and Lipperts, 2002): ''An agent is a specific software entity which acts autonomously, yet on behalf of a user or another agent. It is able to perceive and influence its environment, e.g. the operating system, applications, and other agents by communicating with these instances.''

3. Fault management

A modern computer network is not just a group of computers that are connected together, but it must also meet the users' and the administrators' requirements. Combining different institutions and organisations makes the networks more complex and this growing of complexity creates a high demand on network management services. Previously, the performance issues for networks had been addressed by increasing the

bandwidth. However, this approach cannot be efficiently maintained in the growth of the complexity and heterogeneity of today's networks.

Fundamentally, network management is a service that employs a range of tools and applications to identify existing and potential loss of service, as well as helping the managers in monitoring and maintaining the network. Fault management tools can help to increase the reliability of the network by identifying the fault and then initiate the recovery process to overcome these faults. For example, when an alarm from a node is detected, a technician will be allocated to resolve the fault at the node location. However, by using more advanced tools, the network manager would be able to go many steps further to isolate and correct a software fault from its location. Consequently, the manager can return the network to normal status in a shorter period of time with minimal effort. Kelley (2004) defines most end users' concerns as: "Network management components perform specific tasks to monitor and manage the security, health and availability of various aspects of the network."

Fault management is the process of locating, analysing, fixing and reporting network problems such as collisions, bad packets and other overhead problems. This, in turn, makes the network more efficient and productive. Fault management can save repair costs during detections, isolations and faults correction procedures. It provides more details by identifying a problem and collecting some data about the current state of the network through the management protocols. Then, the manager station can apply first aid to re-establish any services that have been lost. This can be accomplished by deciding if the fault could be managed or isolated and finally corrected. These are the steps of reactive fault management. Moreover, proactive fault management can be prepared for potential faults that might occur in the future, thereby improving the network uptime.

4. Mobile agents in distributed network management

The comparison of the structure of the system, based on intelligent MAs and the current network management systems, adopting a centralized paradigm to manage the network domain system based on SNMP and RMON technology is shown in Fig. 1. The management information is stored in the management information bases (MIB), each device has its own MIB, and the manager station can obtain this information by using the management protocols, for example SNMP. Afterwards, it can apply its functions that probe the network status in order to give clear reports or statistical graphs.

Nevertheless, the centralisation paradigm as shown in Fig. 1a is suitable for the applications with a limited need for distributed control. Fault management includes the discovery, isolation and fixing of problems, the efficiency of fault management is crucial to guarantee the recovery of faults that may occur during the network life cycle. In previous research (Wittner et al., 2000; Hood and Ji, 1997; Pissinov et al., 2000; Li and Baras, 2000; Gurer et al., 1996; Goldszmidt, 1996; Gavalas et al., 2000) in network fault management (NFM), the scalability limitations of centralised network management (NM) is addressed and it becomes significantly more pronounced when transfers of bulk network monitoring data is considered.

Recently, the use of MA technologies as seen in Fig. 1b to automate the fault management functionality in NM has been explored. The MA has the ability to migrate from one node to another in order to read some information from the network elements. Then, the MA can start to investigate the current state of the network. The

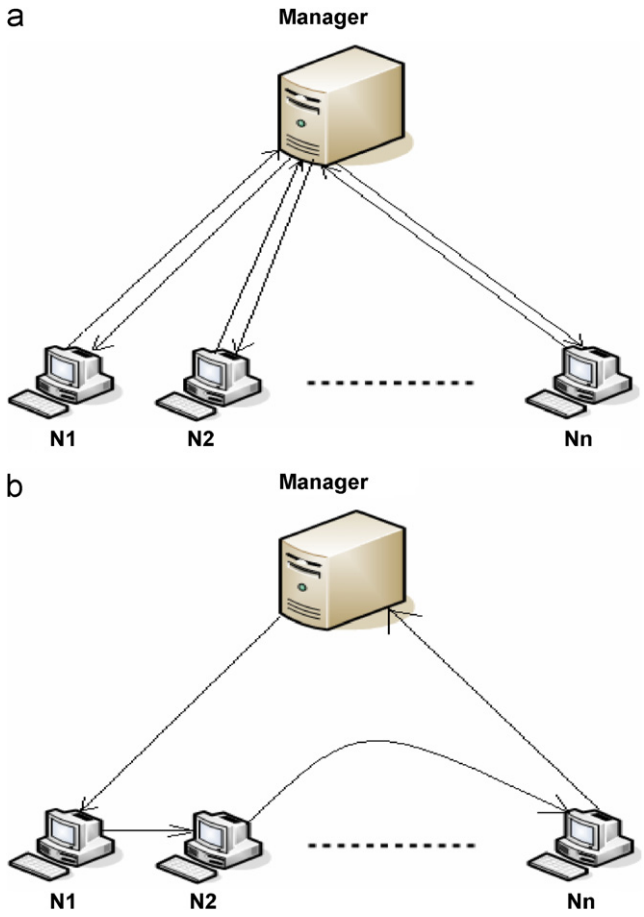

Fig. 1. Comparison of structure of the system based on intelligent MAs (b) with the one of traditional system based on centralized systems (a).

intelligent and portable capabilities can help the MA to perform the latter tasks and send an acknowledgement to the manager node for the purpose of archiving. MA technology can offer a new paradigm for communications over heterogeneous network channels. A number of advantages of using MAs have been proposed and identified. For example: overcoming network latency, educing network load, executing asynchronously and autonomously, adapting dynamically, operating in heterogeneous environments and having a robust and fault-tolerant behaviour (DDRI and diversified data resources).

Potentially, MAs provide better efficiency for the whole system. A client program migrates to a server node, communicates directly with a server programme and returns to an original node with the result. In that approach, the number of remote interactions and the amount of data communicated over the network is reduced; consequently, the network traffic is reduced. Furthermore, better reliability is achieved because the connection between nodes need not be established constantly.

4.1. Communication model

In general, there are four main entities in the area of network management. The first entity is the Top Manager; this has all the functions and the procedures to interact with other entities in order to give the human manager all the details about the domains. The second entity is the Peripheral Devices Data; this provides the data that reflects the current situation of a device. The third entity is the Result Receiver, it could be the Top Manager or any device in the domain. This uses the result, which is generated by the Top Manager, to change its situation from abnormal to normal situation, depending on the management task involved. The fourth entity is the Communication Process, which is the how the Top Manager "talks" with the peripheral devices.

The NM task is taking place as the starting point in the Top Manager. The Top Manager should be attached to a network device that can offer enough computational resources for the management computation purposes. During the management process, the manager needs some stored management data from the peripheral devices individually. According to different management tasks, this data may be requested many times during the management process. After the task process, there is a possibility to have configuration results as consequences of the management process and these results may be required for the Top Manager and the peripheral devices in the domain. Most importantly, if the interaction happens in the same device, then the interaction is a local interaction. If the interaction happens with the Top Manager and another device, the interaction is a remote interaction. These interactions will affect the performance of the NM, with issues like the delay, the increased traffic and the bandwidth consumption.

4.2. Interaction mapping and performance calculation

Before we start with the performance computation and evaluation of the two networking paradigms, we should first describe two of the performance parameters that are used during this study. The first is Network Traffic, such as traffic generated around network devices. The traffic parameters illustrate the overhead generated by the management application to the network. Furthermore, this can be used to determine the bandwidth bottlenecks of the network devices. The second parameter is the Management Time, for example, the time used by the management procedure and the time used by the interactions between different nodes in the domain. These parameters give a clear picture of the response speed of the management applications.

4.3. Performance in client–server paradigm

After addressing the communication model in the previous section and the performance parameters of interest, we need to map the communication model to the specific management paradigm. Herein, we should determine which entities will be the Top Manager, Peripheral Devices Data, Result Receiver or Communication Process and, most importantly, how these entities will interact to accomplish the management task activity. In CS, the task manager is the Top Manager that has all the information and the algorithms and knows how to execute the management task. The MIB at n different nodes are Peripheral Devices Data and these nodes are also the Result Receivers to be reconfigured by the task manager. Finally, the SNMP is the communication method.

The implementation of the performance-monitoring task can be mapped into the CS paradigm as follows: the Task Manager at node n_0 sends out a data query S_q to the node n_1. The SNMP agent that receives the query will access the data from the MIB and it will send Response S_y to the Manager. The Task Manager has many of the management functions which will be applied to the data and may initiate new queries or give some configuration results as a result of that operation. Subsequently, the configuration data will be sent to the node n_1, if it is required. This process is applied for other nodes in the domain.

The average traffic around the node n_0 (manager node) can be calculated as follows:

$$T_{n_0}^{CS} = ((S_q + S_y)\bar{n}_{mib} + S_{con}P_{rcon})n - np + S_q np, \tag{1}$$

where $\bar{n}_{mib}, P_{rcon}, S_{con}, P$ are the average numbers of MIB, the probability of the configuration after the management task has finished, the size of data required to configure a node and the probability of the failed node (Al-Kasassbeh and Adda, 2006), respectively.

Correspondingly, the traffic around a given node n_i is

$$T_{n_i}^{CS} = ((S_q + S_y)\bar{n}_{mib} + S_{con}P_{rcon})p_{in}, \tag{2}$$

where P_{in} is the probability of a node to be involved in the management task, for more information about probability in our study, you can refer to our work in Al-Kasassbeh and Adda (2006).

The average execution time for the task in CS paradigm is

$$t_{Total}^{CS} = \left(\left(\frac{S_q + S_y}{B_W}f\right) + 2t_d + (t_{mib} + t_{proc})\bar{n}_{mib} + \frac{S_{con}}{B_W} + t_d\right)n - np + \left(\frac{S_q}{B_W} + t_d\right)np, \tag{3}$$

where B_w is the bandwidth, t_d is the link delay and $(t_{mib} + t_{proc})$ are the average time for accessing and processing the MIB data.

4.4. Performance in MA paradigm

In MA mode, a MA having the management algorithm, is used as the manager. Instead of carrying all the data back to the node n_0 to run the management task the MA, upgraded with all the management function needed, is sent out to nodes $n_1, n_2, \ldots, n_Q$ in order to access the data and apply the functions locally. In this case, the communication between the MA, with the functions, and the Management Data becomes local interaction. In contrast, the result generated at the managed node should be sent back to the management station or to another node if it is needed to be reported. Moreover, if a configuration is needed at the node, the configuration also becomes a local interaction. If the process needs further data from other nodes, the MA will dispatch itself to the required node where the data is allocated. This process will be repeated till the task is completed. Then, the final result will be sent back to the main station as text or graphs for inspection.

Under this scenario, the average traffic around node n_0 where the task has been started is:

$$T_{n_0}^{MA} = 2S_{MA} + \sum_{i=1}^{Q.p_r} S_{A_i} + \sum_{i=1}^{Q.p_q} S_{r_i}, \tag{4}$$

where the size of the MA is S_{MA}, P_r is the probability to have a report, S_A is the average size of the alert message. Q is the number of the nodes in the domain, S_r is the average size of the

reports that are generated from each node after the MA applied the management functions and P_q is the probability the node will have a report.

Correspondingly, the average traffic around node n_i is

$$T_{n_i}^{MA} = 2S_{MA} + \sum_{j=1}^{i-1.p_q} S_{r_j} + S_A p_q + S_{r_i} p_r. \tag{5}$$

The traffic around node n_i includes getting the MA from some other node and sending the MA to the next node. It also includes the traffic of the result reporting from the same node which may be used in next node.

The average execution time in MA paradigm can be calculated as

$$t_{Total}^{MA} = \text{The transportation time of MA} + (t_{mib} + t_{pross}) + S_r. \tag{6}$$

Herein, we bring in two approaches to poll the data from the domain. The first one is the Accumulative Model and the second is the Interactive Model. The MA in the Accumulative Model (see Fig. 2(a)) travelling from one node to another in the sub-domain, runs the management process, collects the results and travels with the results to the next node. This process is repeated in every node in the management task. In this scenario, the MA is increasing in size which affects the traffic and the delay. This approach is not efficient if the number of nodes, in the domain, is very large. In the Interactive model, as seen in Fig. 2(b), as soon as the MA arrives at the node, it will clone itself. The first MA will reside at the node and the clone migrates to the next node with the same mission task. The MA in the node will start the management process at the end it sends the results to the Top Manager.

For the accumulative model the average execution time $t_{n_i}^{MAA}$ for n_i node can be calculated as

$$t_{n_i}^{MAA} = \frac{S_{MA_i}}{B_{W_{0,1}}} + \left(\frac{S_{MA_i} + \sum_{j=1}^{q} S_{rj}}{B_{W_{i-1,1}}} + (t_{mib} + t_{proc})\bar{n}_{mib} \right) p_r. \tag{7}$$

For the Interactive model average execution time $t_{n_i}^{MAI}$ for the node n_i is

$$t_{n_i}^{MAI} = \left(\frac{S_{MA_i}}{B_{W_{i-1,1}}} \right) p_r + \frac{S_r}{B_{W_{0,i}}} + (t_{mib} + t_{proc})\bar{n}_{mib}. \tag{8}$$

The transportation time of MA is the summation of sending transportation time from n_0 and receiving transportation time from n_Q

$$t_{sen}^{MA} = \frac{S_{MA_i}}{B_{W_{0,1}}} + t_{d_{0,1}}, \tag{9}$$

$$t_{rec}^{MA} = \frac{S_{MA_i} + \sum_{i=1}^{Q_r} S_r}{B_{W_{Q,0}}} + t_{d_{q,0}}. \tag{10}$$

Now, we can calculate the total execution time as

$$t_{Total}^{MA} = T_{sen} + \sum_{i=1}^{Q_r} t_{v_i} + T_{rec}. \tag{11}$$

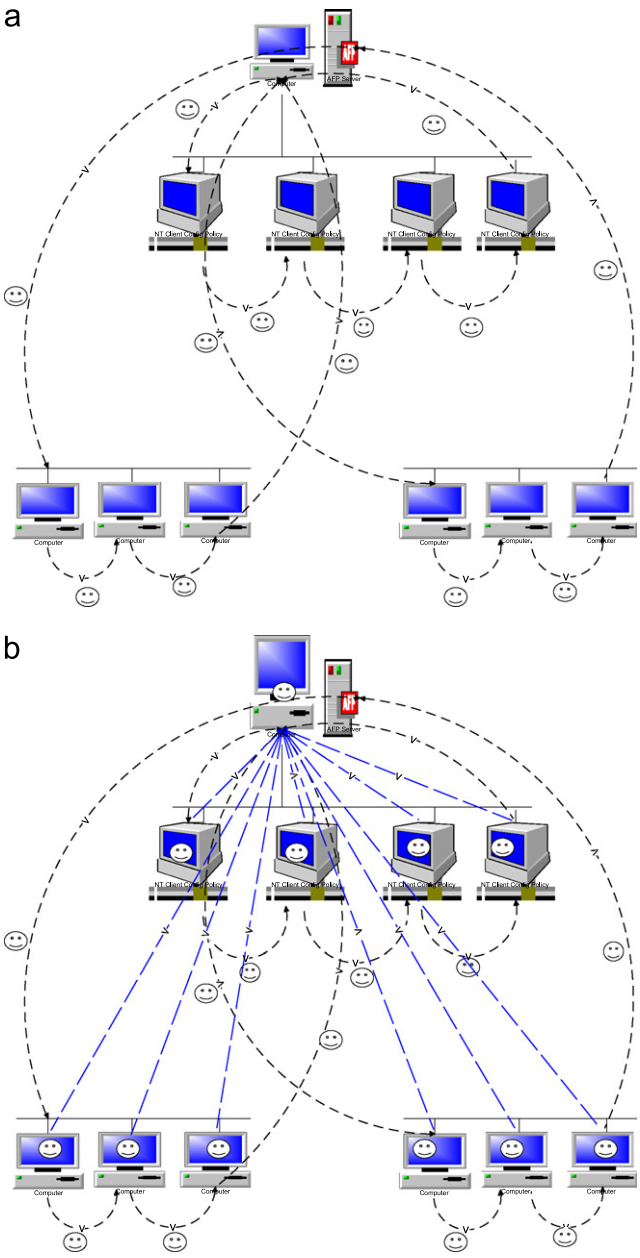

Fig. 2. Approaches to polling: (a) accumulate and (b) interactive.

4.5. Numerical results

In this section, we compare the communication performances of the CS and the MA paradigms from two different aspects—scalability and reliability. Derived from the previous section, we conclude that the scalability of the CS paradigm is primarily affected

by the traffic and the computational load generated around the main station. The traffic around the main station increases with the enlarge of the amount of data (MIB) accessing at the same network element. Alternatively, in the MA paradigm, the traffic load around the management station mostly generates the reporting of the results from the network elements. Furthermore, with the smaller size MA, the traffic around the network management station is reduced considerably; in general the data transferring from the nodes is much higher than the traffic generated by transferring just the results from the nodes. Simultaneously, in the MA paradigm, the workload at the main station is reduced, because the management tasks are executed locally on the nodes. On the other hand, under the MA paradigm the workload locally at the nodes may increase as compared to the CS paradigm. However, in the MA paradigm there is no bottleneck and, as a consequence, it presents a more scalable architecture.

Most NM systems poll the managed node searching for error conditions, and illustrate the problem in either a graphic or a textual interface. The manager node needs to know about the state of the network, so gathering information from the managed nodes is the first step in NM. There are two methods to collect the information from the domain (Leinwand and Conroy). The first one is occasionally polling network devices and the second is logging critical network events. In fact, in some cases, both of them might be used at the same time. The first method can help to discover faults in a timely manner. It gives trustworthy detection of failures and reducing protocol complexity, as well as reducing performance impact on the managed entity. However, the shorter the notification time desired, the greater the amount of consumed bandwidth. The second method is transmitting the alerts from the managed device to the manager if they have any difficulties. This method gives real-time knowledge of the systems difficulties with minimal amount of network management traffic. In the following, we consider a network station that has a number of nodes in its management domain. In order to perform some management task, we presume that the manager station needs to check the MIB objects in every node in the domain consecutively, according to the management algorithms.

The values of the related parameters that we used in our study are as follows: the MIB size of a common status (Request (S_q)/Response (S_y)) size is 28/33 bytes (Park, 2004). The protocol overhead of IP, UDP, and SNMP is 60 bytes altogether for both response and request. Partial result $S_{r,,}$ we assumed is 200 bytes. The reporting probability P_r is 1; the size of the MA depends on its functions, as an average we assumed it to be approximately 3 kbytes. The bandwidth available between any pair of nodes is assumed to be 512 kbps, and finally the number of nodes ranged from 3 to 30.

Fig. 3 compares the traffic generated around the main station in different paradigms for different data requests by using formulas (1) and (4), as we can see the traffic around the main station is nearly fixed in the MA model, no matter what the size of the data from the MIBs is, or how many nodes exist within the domain.

In general, the MA mode performed better than the CS mode. Though, if the extracted data from the MIBs is smaller in size than the MA and the system contains a small number of nodes, the CS paradigm could work better than the MA paradigm because of the MA size.

Fig. 4 shows the comparison of the management reaction time for different management paradigm. Herein, we illustrate the CS model with different MIBs variable numbers, and also for the MA, the accumulative and interactive model is pointed up too. As expected, the total traffic generated and the management reaction time increased as the number of

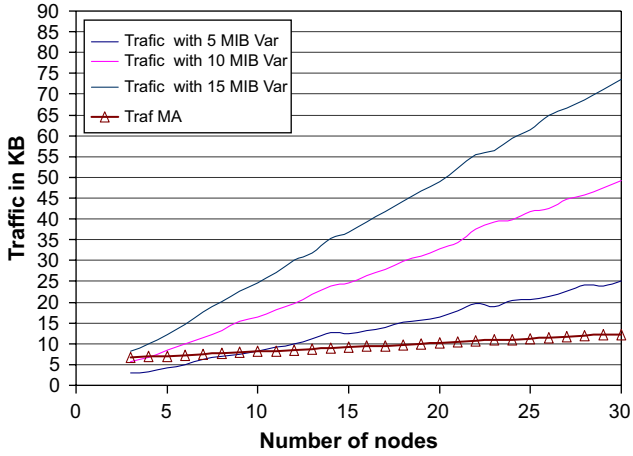

Fig. 3. Traffic generated around the management station.

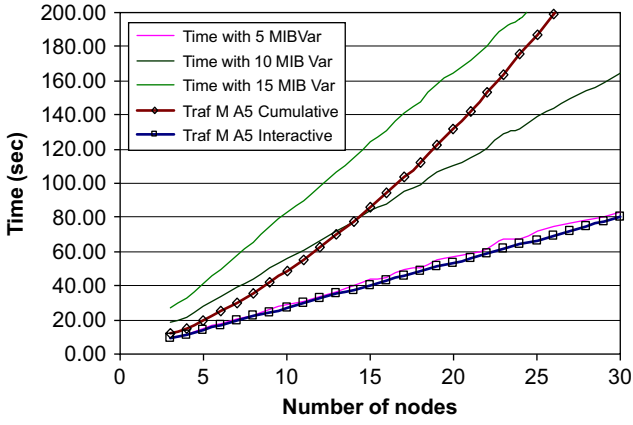

Fig. 4. Management reaction time.

nodes in the domain increased. In addition, the interactive model is out-performing the accumulative model and it is very useful for the real time domain.

4.6. Experimental results

In order to evaluate the performance of MA and CS in real-time, an experiment has been conducted on the local area network (LAN) of the School of Computing at University of Portsmouth. In our experiment, we dispatch a MA to compute the utilisation of the network card of 15 machines over 10 s period of time. These workstations belong to different subnets, and run various operating systems, such windows XP, Windows 2000 and Linux.

A fully documented of the MIBs variable used in our calculation for utilisation can be found in Adhicandra et al. (2003). The access to the MIBs of each workstation has been

obtained by AdventNet SNMP package (AdventNet). The MAs are built round the aglet developed by IBM (Lange and Oshima, 2003), and the traffic crated by the MA and CS paradigms has been analysed by Ethereal traffic analyser (Combs, 2004).

Fig. 5 compares the management reaction time in MA and CS paradigms. It shows that the CS and the accumulative approaches are very similar, but the interactive approaches takes less time, because all the agents process the MIBs concurrently using cloning. In our experiment, the MA have to wait 10 s to take the initial and final MIBs values, due to the high speed of our network, the process and travelling time is very little.

Fig. 6 shows the traffic generated around the management station in the network, as we can see the CS supersedes the MA if the number of the nodes is small, but the MA consume the same amount of traffic even the number of nodes are increasing, not like the CS, the traffic is directly proportional to number of the nodes. This leads to having a hybrid model, the MA can use both techniques, centralised and decentralised to

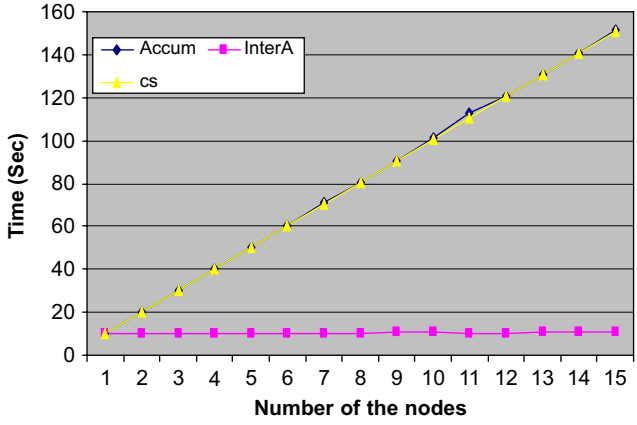

Fig. 5. Management reaction times form the experiment.

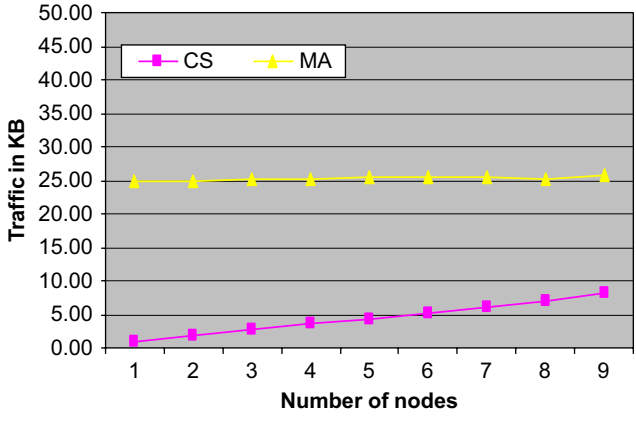

Fig. 6. Traffic generated around the management station in the real network.

accomplish its tasks, MA is very efficient if we have remote subnets then one agent will despatch to those subnets to do the entire task and it will just send the report to the manager. Consequently, we proposed Adaptive Intelligent Mobile Agent that can use these two techniques based on its knowledge about the domain. The MA can be like CS in small subnets and can be distributed in remote and large subnets.

The performance of MA depends on its protocol and its size. As the experiment has shown, aglet realizes on ATP protocol which itself based on HTTP, thus more traffic needed to establish connections between the manager and polled workstations is induced. This is essential to mention secure and reliable agent. On the other hand, CS uses UDP protocol, which involves less traffic.

5. Conclusion

In future networks, it will be impractical to send a technician to test every faulty node because there will, potentially, be too many to cover. An MA with enough knowledge about the problem may run a number of tests and attempt to repair the node remotely, mainly if it is a software fault. In this paper, a standard framework that can be used for the evaluation and analysis of the performance of the MA management paradigm is presented and analysed. Although the importance of this paper is placed on the calculation of the possible performances under the MA-based management approach, the framework is used to gain some idea about how to use different management architectures under different networking management tasks. We present some numerical and experimental results for different networking scenarios that demonstrate the applicability of Adaptive Intelligent Mobile Agent.

References

Adhicandra I, Pattinson C, Shaghouei E. Using mobile agents to improve performance of network management operations, 2003.

AdventNet S. API 3.0.

Al-Kasassbeh M, Adda M. The architecture of the intelligent mobile agent in network fault management. In: Proceedings of international conference on computer science and information systems, Athens, 2006.

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.

Carzaniga A, Picco GP, Vigna G. Designing distributed applications with mobile code paradigms. In: Proceedings of international conference on software engineering, vol. 19; 1997. p. 22–33.

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

Combs G. Ethereal: a network protocol analyzer, 2004.

DDRI and Diversified Data Resources. Web based SNMP network management system an introductory overview of SNMP.

Gavalas D, Greenwood D, Ghanbari M, O'Mahony M. Advanced network monitoring applications based on mobile/intelligent agent technology. Comput Commun 2000;23:720–30.

Goldszmidt G. Network management views using delegated agents, 1996.

Gurer DW, Khan I, Ogier R. An artificial intelligence approach to network fault management. Menlo Park, CA: SRI International; 1996.

Hood CS, Ji C. Proactive network fault detection, April 1997.

Kelley D. Does your wired network know what your wireless network is doing? 10 December 2004.

Lange DB, Oshima M. Programming and deploying Java mobile agents with aglets. Addison-Wesley, 2003.

Leinwand A, Conroy KF. Network management—a practical perspective, 2nd ed. Addison-Wesley Publishing Company, Inc.

Li H, Baras JS. Intelligent distributed fault management for communication networks, TR 2000.

Liotta A. Towards flexible and scalable distributed monitoring with mobile agents. In: Department of Computer Science. vol. Degree of Doctor of Philosophy London: University of London, 2001.

Nikaein N. Reactive autonomous mobile agent, 1999.

Park AS-B. A service-based agent system supporting mobile computing. In: Heinisch-Westfalischen Technischen Hochschule Seoul (Süd-Korea), 2004.

Pissinov, N., Bhagyavati, Makki K. Mobile agents to automate fault management in wireless and mobile networks. Lecture notes in computer science, 2000. p. 1296–1300.

Pleisch S, Schiper A. Approaches to fault-tolerant and transactional mobile agent execution—an algorithmic view. Comput Surveys 2004;36(3):219–62.

Richard S, Lipperts G. Mobile agent support services. In: Department of Mathematic, Informatics and Nature Sciences: Rheninsch-westflischen technischen hochschule Achen, 19 February 2002.

Wittner O, Helvik BE, Hoepler CJE. Failure semantics of mobile agent systems involved in network fault management. In: Networked planet: management beyond 2000; NOMS 2000, 2000. p. 939–40.